

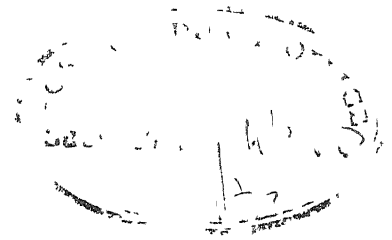
# **EFFECTIVE ALGORITHMS FOR THE CYCLIC STAFF SCHEDULING PROBLEM**

**A Thesis Submitted  
In Partial Fulfilment of the Requirements  
for the Degree of  
MASTER OF TECHNOLOGY**

**By  
S. V. RAMA REDDY**

**to the  
INDUSTRIAL & MANAGEMENT ENGINEERING PROGRAMME  
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR  
MARCH, 1985**

## CERTIFICATE



This is to certify that the work embodied in the thesis entitled, "Effective Algorithms for the Cyclic Staff Scheduling Problem," by S.V. Rama Reddy has been carried out under our supervision and has not been submitted elsewhere for the award of a degree.

A handwritten signature in black ink, appearing to read "R. K. Ahuja", with a horizontal line drawn underneath it.

(R. K. Ahuja)  
Lecturer  
Industrial and Management Engg.  
Indian Institute of Technology  
Kanpur 208016, INDIA

A handwritten signature in black ink, appearing to read "J. L. Batra", with a horizontal line drawn underneath it.

(J.L. Batra)  
Professor and Head  
Industrial and Management Engg.  
Indian Institute of Technology,  
Kanpur 208016, INDIA

March, 1985

18 JUN 1985

17 2216

100000

87610

## ACKNOWLEDGEMENTS

I express my deep sense of gratitude to my thesis supervisors, Dr. R.K. Ahuja and Dr. J.L. Batra, for the responsible and stimulating guidance and profuse assistance I have received from them throughout the course. Indeed, they are responsible for taking out the best in me.

I am thankful to my friend, Mr. V.V.S.N. Murty, for his help in many ways. I am also thankful to Mr. S.S.R. Murty, Mr. M.S. Suryanarayana and other friends for their ready willingness to help me at any time.

I thank Mr. J.K. Misra for his excellent typing, Mr. Bajpai for his appreciable plotting of charts, and Mr. Buddhi Ram Kandiyal for his immaculate cyclostyling work.

S.V. Rama Reddy



## CONTENTS

Chapter	Page
I. INTRODUCTION	1
1.1 Introduction	1
1.2 Preliminaries	5
1.3 Statement of the Problem	6
1.4 Literature Review	8
1.5 Outline of the Thesis	15
II. LP-BASED HEURISTIC ALGORITHM	18
2.1 Introduction	18
2.2 Development of the Heuristic Algorithm	18
2.3 Statement of the LP-Based Heuristic Algorithm	22
2.4 Numerical Example	23
III. EXACT ALGORITHMS	27
3.1 Introduction	27
3.2 Notations	28
3.3 Branch and Bound Algorithm	29
3.3.1 Bounding Strategy	29
3.3.2 Fathoming Strategy	30
3.3.3 Pruning Strategy	30
3.3.4 Branching Strategy	30
3.3.5 Search Strategy	32
3.3.6 Sensitivity Analysis	32
3.3.7 Numerical Example	38
3.4 Cutting Plane Algorithm	40
3.4.1 Introduction	40
3.4.2 Development of the Algorithm	41
3.4.3 Source Variable Selection	44
3.4.4 Statement of the Algorithm	46

<u>Chapter</u>	<u>Page</u>
IV. COMPUTATIONAL INVESTIGATIONS	48
4.1 Introduction	48
4.2 Problems Considered for Experimentation	48
4.3 Computational Performance	52
4.4 Concluding Remarks	55
REFERENCES AND BIBLIOGRAPHY	64
APPENDIX	

## CHAPTER I

### INTRODUCTION

#### 1.1 Introduction:

In recent years, much interest has been focussed on various problems that arise in cyclic staffing. Such problems involve the optimal scheduling of resources to meet demands where both resource availability and demand profile are cyclic.

Many industries and public services operate around the clock, 7 days a week. This is the case in police and fire departments, in telephone companies, in hospitals and in many heavy industries. Typical work schedules for employees of such services and industries normally consist of a cycle spreading over a few weeks and in which shift work and holiday periods alternate. These schedules consist of a cycle which repeats itself after a few weeks. Employees can therefore be seen as continuously going **through** the same basic work-holiday pattern or rotating schedule. One rotating schedule is determined by the length of its cycle, by the length and frequency of the work and rest periods and by the way these alternate.

The objective in cyclic staff scheduling problem is to minimize the total labor hours needed to meet demand requirements during different time periods. This problem is efficiently solvable when resources are continuously available during consecutive time periods. For an example of continuous availability,

let us consider one fundamental problem, called (5-7) cyclic staffing problem. In this problem, both resource availability and requirements are cyclic. The objective is to minimize the number of workers needed to meet daily requirements, with the added provision that each worker be allowed two consecutive days off each week.

Let  $x_j$  denote the number of workers in shift  $j$  and  $b_i$  denote the requirement during time period  $i$ . With these notations the above problem can be formulated as:

$$\begin{array}{ll}
 \text{Minimize} & \sum_{j=1}^7 x_j \\
 \text{s.t.} & \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} \geq \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} \\
 & x_j \geq 0 \text{ and integer for all } j = 1, \dots, 7.
 \end{array}$$

Here the work-shift is five consecutive days. If we denote the 0-1 matrix by  $A$ , each column of  $A$  represents a possible work shift and each row of  $A$  shows the shifts that are active on the corresponding day. That the resources are continuously available is reflected in  $A$ , where each column contains exactly

one block of consecutive ones (when viewed circularly). This problem has been solved optimally by Bartholdi et al [6].

In a more realistic model, resources are intermittently available. This may be due to, for example, lunch and relief breaks for the workers. The following matrix is an hypothetical example showing intermittently available resources.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

This is an (8 x 8) matrix and shift length is five. It can be observed that there is one break (when viewed circularly), i.e. two blocks of consecutive ones, in each row.

Bartholdi [5] showed that the problem of cyclic staff scheduling with intermittently available resources is a NP-complete problem. In view of this observation, the possibility of developing an exact algorithm for the above problem is remote. Heuristic approaches based on linear programming relaxation, network relaxation or lagrangean relaxation are available in literature. In all of them, some explicit use was made of the fact that  $A$  almost had the property of consecutive ones.

Bartholdi [5] developed an algorithm based on linear programming round-off by making no assumptions about the circularity of A. He gives a bound on the maximum possible deviation of his heuristic from the integer optimum solution. This bound is not good for the problems in which intermittent breaks are reasonably large in each shift. More often than not, it is observed that the deviation of the solution obtained using Bartholdi's method from the integer optimum solution is almost equal to the bound given by him.

In this dissertation, we consider the problem of scheduling with intermittently available resources and develop some methods for solving it. We develop three algorithms. One is LP-based heuristic algorithm. In this algorithm, a sequence of LP's are solved and an improved version of the rounding-off procedure suggested by Bartholdi et al [6] is used after solving each LP to get a feasible solution to the cyclic staff scheduling problem. The other two algorithms are Branch and Bound algorithm and cutting plane algorithm. These two are exact algorithms. Branch and Bound algorithm makes use of sensitivity analysis to impose lower or upper bound constraints on any variable and LP-based heuristic is applied at every node in the branching tree. The cutting plane algorithm makes use of the dual fractional cutting plane algorithm. Two types of cuts are introduced to reduce the total number of cuts needed.

All the suggested algorithms have been programmed and tested extensively on randomly generated cyclic staff scheduling problems. The results are quite encouraging and reported. The solutions obtained using LP-based heuristic algorithm are observed to be very close to the optimum solution for all the problems solved. The Branch and Bound algorithm is able to solve problems of size  $(96 \times 96)$  optimally in a few minutes of CPU time. The performance of cutting plane algorithm was observed to be erratic. It could not solve problems of size more than  $(48 \times 48)$ .

## 1.2 Preliminaries:

Some notations and well-known concepts of cyclic staff scheduling problem are used throughout the thesis. For the sake of completeness they are given below.

A 0-1 vector is said to be circular if its 1's occur consecutively, where the first and last entries are considered to be consecutive.

A matrix is called column circular (resp., row circular) if its columns (resp., rows) are circular.

The term scheduling in this context, denotes the act of specifying how many resources should there be in different shifts so as to satisfy the requirements during different time periods with minimum cost.

Following notations are used in the thesis:

- $m$  total number of time intervals to be scheduled.
- $n$  total number of different shifts or work schedules.
- $A$  an  $(m \times n)$  matrix of 0's and 1's.
- $x_j$  number of operators assigned to the  $j$ -th work schedule.
- $b_1$  number of operators required for time interval 1.
- $a_{ij} = \begin{cases} 1 & \text{if time interval } i \text{ is a work period in the } j\text{-th} \\ & \text{work schedule.} \\ 0 & \text{otherwise.} \end{cases}$
- $c_j$  cost of assigning a worker to shift  $j$ .
- $q$  maximum number of blocks of consecutive ones (when considered circularity) in any row in matrix  $A$ .
- $k$  minimum number of ones in each column of  $A$ .
- $X$  assignment vector  $(n \times 1)$ .
- $C$  cost vector  $(1 \times n)$ .
- $b$  requirement vector  $(m \times 1)$ .
- $O$  vector of zeros  $(m \times 1)$ .
- $1$  vector of ones  $(1 \times n)$ .
- $\lfloor . \rfloor$  the greatest integer less than or equal to ' $.$ ' .
- $\lceil . \rceil$  the least integer greater than or equal to ' $.$ ' .

### 1.3 Statement of the Problem.

The problem of cyclic staff scheduling can be posed as follows: given resource requirements for each time period within



a planning horizon, find out a schedule of resources to meet these requirements at minimum cost. This can be formulated as an ILP as given below.

$$\begin{aligned}
 \text{Min.} \quad & \sum_{j=1}^n c_j x_j \\
 \text{s.t.} \quad & \sum_{j=1}^n a_{ij} \cdot x_j \geq b_i, \quad i = 1, \dots, m \\
 & x_j \geq 0, \quad j = 1, \dots, n \\
 & \text{and integer.}
 \end{aligned}$$

This can alternatively be stated as,

$$\begin{aligned}
 \text{Min.} \quad & C X \\
 \text{s.t.} \quad & A X \geq b \\
 & X \geq 0 \text{ and integer,}
 \end{aligned}$$

where  $C$ ,  $b$  are integral vectors and  $A$  is an  $m \times n$  matrix with all entries either zero or one. Without loss of generality we assume that  $b > 0$ ,  $C \geq 0$ , and that  $A$  has atleast one 1 in each row. Each  $b_i$  may be imagined to represent the number of workers required during time period  $i$ , and each  $c_j$  the cost of assigning a worker to shift  $j$ . Each column of  $A$  may be imagined to represent a possible work shift.

In this dissertation, we consider a special case of the above problem that is  $c_j = 1$ ,  $\forall j = 1, \dots, n$ . The problem then is to determine smallest number of workers to satisfy the demand schedule.

Mathematically the problem is.

Minimize  $1X$

s.t.

$$A X \geq b$$

$$X \geq 0 \text{ and integer.}$$

Subsequently this problem is referred to as cyclic staff scheduling (CSS) problem.

#### 1.4 Literature Review:

The cyclic staff scheduling problem has been studied by several researchers in the literature. Dantzig [7] formulated the general CSS problem as an ILP as given below.

$$\text{Min. } \sum_{j=1}^n c_j X_j$$

s.t.

$$\sum_{j=1}^n a_{ij} X_j \geq b_i, \quad i = 1, \dots, m$$

$$X_j \geq 0, \text{ integer, } j = 1, \dots, n.$$

where  $X_j$  is the number of workers in shift  $j$ , and  $C_j$  is the cost of assigning a worker to shift  $j$ ,  $a_{ij} = 1$  if shift  $j$  is present on  $i$ -th time period and  $a_{ij} = 0$  otherwise. The  $b_i$  is the requirement during time period  $i$ .

Henderson [9] has considered a special case of Dantzig's general ILP formulation for scheduling of telephone operators as given below.

$$\begin{array}{ll}
 \text{Min.} & \sum_{j=1}^n X_j \\
 \text{s.t.} & \\
 & \sum_{j=1}^n a_{ij} X_j \geq b_i, \quad i = 1, \dots, m \\
 & X_j \geq 0, \text{ integer, } j = 1, \dots, n
 \end{array}$$

Luce [12] has suggested an alternative approach for scheduling of telephone operators. He essentially modified the ILP formulation of the CSS problem suggested by Henderson by adding shortage and surplus tolerance variables. Luce's ILP problem is more difficult than solving Henderson's problem as more integer variables are needed for formulation. However, a constraint fixing the total number of operators can be added in Luce's model. The underlying assumption of Luce's model is that "cost" of deficit equals "cost" of surplus. Penalizing under-staffing and over-staffing equally is indeed a little peculiar and unrealistic.

The cyclic staff scheduling problem is more easily solvable when resources are continuously available. Baker [3] has suggested a method which needs only hand computations to solve the (5-7) cyclic staffing problem optimally. In this problem, full-time workers are available for five consecutive days in a one week cycle with two consecutive days off.

Baker [4] has suggested another simple algorithm for the problem of allocating staff in order to meet demand over a

repeating seven-day cycle when both full time and part-time workers are there. The formulation assumes that full-time employees are entitled to two consecutive days off and that the objective is to meet demand for staff with a minimum use of part-time employees.

For solving the general cyclic staffing problem with continuously available resources, Bartholdi et al [6] suggested an optimal technique based on linear programming round-off. In this method, the given CSS problem is first solved as an LP by relaxing the integrality restriction on all the variables. The continuous valued optimum solution obtained is then rounded-off in a special way, which guarantees the feasibility and optimality of the rounded-off solution to the original problem.

In practice, resources may not be available continuously. This may be due to lunch or relief breaks for the workers. The CSS problem with intermittently available resources is more difficult to solve. Current applications resolve the dilemma of problem size in a variety of ways. One possibility is to solve the CSS problem as an LP and then to use some heuristic procedure to round-off the resulting non-integer solution.

The scheduling of intermittently available resources has been studied by Glover and Mulvey [8], Segal [18], Henderson and Berry [10], Laporte [11], Shepardson and Marsten [19], Bartholdi [5], Ondhensden and Wen-Jang [16], Willis and Wilson [21], Mabert and Watts [13], and Morris and Showalter [14].

Segal [18] considered the problem of scheduling telephone operators where operators work in shifts that are specified by start and end times. Each tour contains from one to three relief periods during which the operators are not working, and a particular schedule of relief periods within a tour is called a trick. He proposes a method for determining the assignment of operators to each trick, it utilizes the out-of-Kilter algorithm and two network-flow models. The problem is formulated as an ILP.

$$\begin{aligned}
 \text{Min.} \quad & \sum_{j=1}^n C_j x_j \\
 \text{s.t.} \quad & \sum_{j=1}^n a_{ij} x_j \geq b_i, \quad \forall i = 1, \dots, m \\
 & x_j \geq 0, \text{ integer}, \quad \forall j = 1, \dots, n.
 \end{aligned}$$

He considers this problem unsolvable with the available integer programming techniques at that time. He follows an approach based on network-flow for formulation in which integer solutions are guaranteed. Initially breaks and relief periods are ignored and the simplified problem is solved using out of Kilter algorithm. He then brings in relief and break periods and uses some heuristic procedure to meet shortage at break and relief periods.

Henderson and Berry [10] gave some heuristic methods for scheduling of telephone traffic exchange operators to meet demand that vary over a 24-hour operating period. Two types of heuristics are described:

- (1) for determining the work shift types to be considered in preparing an operator shift schedule and,
- (2) for constructing an operator shift schedule from a given set of work shift types.

They give two heuristics for determining the set of shift types. Once the working subset (shifts) has been defined, the given CSS problem is solved as a relaxed LP and all the variables are rounded-up and then two heuristics are applied to remove the excess operators.

Bartholdi [5] has suggested a guaranteed accuracy round-off algorithm for solving the CSS problem with intermittently available resources. We describe his method somewhat in detail in a separate section.

Oudheusden and Wen-Jenq [16] consider the following problem, given operator requirements during different time periods of the day, a certain number of operators are assigned to the particular work schedules in order to meet, as well as possible, the varying demand for operators. If the number of operators for a day are fixed at  $p$ , the following model is suggested,

Max.  $Y$

s.t.

$$\begin{aligned}
 & \left( \sum_{j=1}^n a_{ij} X_j - b_i \right) \geq Y, \quad i = 1, \dots, m \\
 & \sum_{j=1}^n X_j = p \\
 & X_j \geq 0, \text{ integer}, \quad j = 1, \dots, n.
 \end{aligned}$$

$Y$  is an integer variable which can be positive, zero or negative.

This is a maxmin problem as it maximizes the smallest difference between supply and demand in any period. When  $Y^{opt}$  (optimum objective function value), takes the value  $-1$ , at least one time interval will be understaffed by one operator. When  $Y^{opt}$  equals  $0$ , all demands are met and overstaffing in all periods is not possible. When  $Y^{opt}$  equals  $+1$ , all operator requirements are met with atleast one extra worker. Results of one realistic study, applied to Taiwan Telecommunication administrative Bureau are presented.

Arthur and Ravindran [2] consider a problem of scheduling nurses under multiple objectives. They make use of a goal programming model which allows for consideration of multiple conflicting objectives inherent in scheduling a nursing staff.

Willis and Wilson [21] present a case study of scheduling telephone betting operators. They demonstrate the application of linear programming and network flow techniques for determining optimal shift sizes for telephone betting operators. Mabert and Watts [13] present a simulation analysis of tour shift construction procedures. They provide some managerial suggestions on the issues of effectively staffing personnel on a weekly basis.

Morris and Showalter [14] give an exact and a heuristic algorithms for solving the CSS problem with intermittently available resources. First they consider the problem of

scheduling workers on a day, where the workers work for 8 hours with a one hour break after 4 hours of work in 24-hour cycle. For solving this  $(24 \times 24)$  problem optimally, they suggest a mixed cutting plane and branch and bound algorithm. They give some heuristic approaches for solving the weekly scheduling problem of  $(168 \times 168)$  size.

#### 1.4.1 Bartholdi's Method.

Bartholdi [5] has suggested a heuristic algorithm to find a feasible solution for the CSS problem when there are intermittent breaks in the shift. In this section, we describe his method in detail.

Let  $A$  be an arbitrary  $m \times n$  0-1 matrix whose  $i$ -th row is  $A_i$  and whose  $j$ -th column is  $A_j$ . Suppose each column of  $A$  has atleast  $K$  ones but no row has more than  $q$  blocks of consecutive ones (when considered circularity). A vector whose every entry is  $(q - 1)$  is denoted by  $(q-1)$ .

The algorithmic steps involved in his method are given below.

Step 1: Solve the continuous-valued linear program  $LP(b+q-1)$ :

$$\begin{aligned} \text{Min. } & 1^T X \\ \text{s.t. } & \\ & AX \geq b + q - 1 \\ & X \geq 0 \end{aligned}$$

Let the solution be  $x_{LP}(b+q-1) = (x_1, x_2, \dots, x_n)$ .



Step 2: Construct the (integer-valued) heuristic solution,

$$x_h = (\lceil x_1 \rceil, \lceil x_1 + x_2 \rceil - \lceil x_1 \rceil, \dots, \lceil \sum_{h=1}^n x_h \rceil - \lceil \sum_{h=1}^{n-1} x_h \rceil).$$

Bartholdi has proved that the solution obtained after rounding-off as described above is feasible to the following problem.

Min  $1^T X$

s.t.

$$A X \geq b$$

$$X \geq 0, \text{ integer}$$

He also obtains an upper bound on the possible deviation of his heuristic solution ( $Z_h$ ) from the integer optimum solution  $Z_{IP}$  to the staffing problem. This bound is,

$$Z_h - Z_{IP} \leq \lceil \frac{n}{k} (q - 1) \rceil.$$

It might be noted that this bound is not satisfactory for higher values of  $q$ .

### 1.5 Outline of the Thesis:

A brief chapter by chapter outline of the thesis is given in this section.

In Chapter II, we develop a heuristic algorithm for the CSS problem. In this method, the given staffing problem is solved as an ordinary LP by relaxing integrality restriction on all variables. An improved version of the special rounding-off procedure suggested by Bartholdi et al [6] is applied. If no

constraint is violated with the rounded-off solution, this solution is optimum. Otherwise, we find out the unsatisfied demand in each time period. This demand is taken as the requirement of the new CSS problem. This is repeated until there is no unsatisfied demand for every time period. The solution obtained using this algorithm is found to be very close to the optimum solution. A numerical example is presented to illustrate the algorithm.

In Chapter III, we develop an adaptation of the general integer programming branch and bound algorithm and a dual fractional cutting plane algorithm for the CSS problem. Some of the important features of this algorithm are.

- (i) Only one A-matrix of fixed size is needed to be stored throughout the problem,
- (ii) Reoptimization is greatly used to reduce the computations substantially,
- (iii) To impose lower (upper) bound type of constraints on any variable, sensitivity analysis of the lower (upper) bound is performed,
- (iv) The heuristic developed in Chapter II is applied at every node of the branch and bound tree to obtain a good feasible solution,
- (v) The variable selected for branching is such that the pruning is more effective.

Detailed discussion of all these aspects is presented. A numerical example is presented to clarify the above steps.

We also develop a cutting plane algorithm for finding an optimum solution to the CSS problem. Dual fractional cutting plane algorithm is used. Two types of cuts are introduced. It is known that in an integer optimum solution to the CSS problem, sum of all variables must be an integer, so whenever the objective value is found to be fractional, we apply the cut that the sum of all original variables must be greater than or equal to the least integer greater than the objective function value. If the objective function value is fractional one, we then apply Gomory's fractional cut. Detailed discussion about the strategy used for finding source variable for cut generation is also given.

In Chapter IV, we present the computational performance of Branch and Bound algorithm, LP-based heuristic, cutting plane algorithm and Bartholdi's method. The results are found to be quite encouraging. The solutions obtained using LP-based heuristic were found to be very close to the optimum solution. The number of LPs solved in any problem were never more than 2. The Branch and Bound algorithm is able to solve problems of size (96 x 96) optimally in a few minutes of CPU time. The performance of cutting plane algorithm was observed to be erratic. It could not solve problems of size more than (48 x 48). Computer programs for all these algorithms were written in FORTRAN-IV and implemented on DEC-1090 Computer System.

## CHAPTER II

### LP - BASED HEURISTIC ALGORITHM

#### 2.1 Introduction:

In this chapter, a new heuristic algorithm for solving the CSS problem is developed. The algorithm is called LP-based heuristic algorithm as a sequence of LP's are solved to get a good feasible solution to the CSS problem. We first describe the intuitive ideas that led to its development. A stepwise description of the algorithm is then presented. A numerical example is solved to illustrate the algorithm.

#### 2.2 Development of the Heuristic Algorithm:

Bartholdi [5] developed a heuristic algorithm for solving the CSS problem. He gave a priori bound for the maximum possible deviation of his heuristic solution ( $Z_h$ ) from the integer optimum solution ( $Z_{IP}$ ) for the CSS problem. The bound is

$$Z_h - Z_{IP} \leq \lceil \frac{n}{k} (q - 1) \rceil .$$

His algorithm does not give a good solution for the CSS problem when there are large number of intermittent breaks (i.e., Bartholdi's  $q$  is large) and/or when  $n/k$  ratio is large. For example, if we consider a (96 x 96) problem with  $k = 30$  and  $q = 6$ , Bartholdi's bound is  $\lceil \frac{96}{30} (6-1) \rceil = 16$  units. Further, it was observed by us computationally that his algorithm

invariably requires as many additional units as given by his bound because he is adding  $(q-1)$  additional units to the actual demand in each time period in the initial stage to obtain a feasible solution to the original CSS problem. We observed that there is no need to add  $(q-1)$  units to the requirement in each time period in the initial stage itself without examining whether it is really needed or not. Our strategy is to provide the additional number of workers on those days where the need arises after applying the special rounding-off to the first LP solution obtained. With this intuitive idea in mind, we develop the LP-based heuristic algorithm for solving the CSS problem.

Familiarity with the following special way of rounding-off is needed before going through the formal description of the algorithm. Given  $n$  continuous valued variables say  $x_1, x_2, \dots, x_n$ , they can be rounded-off in the following special way suggested by Bartholdi et al [6].

$$Y = ([x_1], [x_1+x_2] - [x_1], \dots, [\sum_{j=1}^n x_j] - [\sum_{j=1}^{n-1} x_j]) \quad (2.2.1)$$

Bartholdi has shown that if a CSS problem is solved as an ordinary LP and the continuous  $n$  values obtained are rounded-off in the above fashion, there will not be more than  $(q-1)$  units of unsatisfied demand on any of the time periods. We generalize this result to the  $n$  integer solutions obtained by the following round-offs.

$$Y^1 = ([x_1], [x_1+x_2] - [x_1], [x_1+x_2+x_3] - [x_1+x_2], \dots, [\sum_{j=1}^n x_j] - [\sum_{j=1}^{n-1} x_j]) \quad (2.2.1.1)$$

$$Y^2 = ([\sum_{j=1}^n x_j] - [\sum_{j=2}^n x_j], [x_2], [x_2+x_3] - [x_2], \dots, [\sum_{j=2}^n x_j] - [\sum_{j=2}^{n-1} x_j]) \quad (2.2.1.2)$$

.

.

.

$$Y^n = ([x_n+x_1] - [x_n], [x_n+x_1+x_2] - [x_n+x_1], [x_n+x_1+x_2+x_3] - [x_n+x_1+x_2], \dots, [x_n+x_1+x_2], \dots, [x_n]) \quad (2.2.1.n)$$

where,  $x_j = x_{j-n}$ , if  $j > n$ .

To find-out whether a given solution is feasible to the given CSS problem or not we find out the unsatisfied demand for each time period with that solution. If the unsatisfied demand on all the time periods is zero, the given solution is feasible to the CSS problem. Let  $Y^h = \{y_1^h, y_2^h, \dots, y_n^h\}$  denote the rounded-off solution vector corresponding to the  $h$ -th rounded-off solution, where  $y_1^h$  denotes the rounded-off value of the variable  $x_1$  in the  $h$ -th solution vector. Let  $U^h = \{u_1^h, u_2^h, \dots, u_m^h\}$

denote the vector of unsatisfied demands when  $Y^h$  is taken as the solution for the CSS problem.  $u_1^h$  denotes the unsatisfied demand on 1-th time period when  $Y^h$  is taken as the solution for the CSS problem. Each  $u_1^h$  is obtained as given below:

$$u_1^h = \text{maximum} \{ 0, b_1 - \sum_{j=1}^n a_{1j} \cdot y_j^h \} \quad (2.2.2)$$

$$\forall i = 1, \dots, m, \quad \forall h = 1, \dots, n$$

Now, we explain how the LP-based heuristic proceeds to get a good feasible solution to the CSS problem. We first ignore the integrality restriction on all the variables and solve an LP to get continuous valued optimum solution to the CSS problem. Now, n-rounded-off solutions  $Y^1, Y^2, \dots, Y^n$  and the corresponding unsatisfied demand vectors  $U^1, U^2, \dots, U^n$  are obtained. Total unsatisfied demand for solution  $Y^h$  is given by  $\sum_{i=1}^m u_i^h$ . We calculate the total unsatisfied demand for each of the n solutions obtained and select the one which gives the minimum amount of total unsatisfied demand. Let this solution be  $Y^p$ . If the total unsatisfied demand for this solution is zero, we have a feasible solution to the CSS problem and stop. Otherwise, a new LP is formulated by taking the demand vector for the given CSS problem as  $U^p$ . We now solve this LP and obtain n rounded-off solutions and select the one with minimum amount of total unsatisfied demand. The values of all  $y_j^p$ 's, of the best solution obtained here are added to the corresponding rounded-off  $y_j^p$ 's

obtained upto the previous LP. This process is repeated until the total unsatisfied demand becomes zero.

A formal statement of the algorithm is given in the next section.

### 2.3 Statement of the LP-Based Heuristic Algorithm:

Step 0: Set  $X^* = 0$ .

Step 1: Let  $X$  be the solution of the following continuous valued variable CSS problem.

Min.  $1^T X$

s.t.

$$A X \geq b$$

$$X \geq 0$$

Step 2: Obtain the  $n$  round-off integer solutions  $Y^1, Y^2, \dots, Y^n$  using (2.2.1.1) to (2.2.1.n). Compute the vectors of unsatisfied demands  $U^1, U^2, \dots, U^n$  using (2.2.2) corresponding to  $Y^1, Y^2, \dots, Y^n$  respectively. Let  $Y^p$  be the solution for which

$$\sum_{i=1}^m u_i^p \leq \sum_{i=1}^m u_i^j, \quad \forall j = 1, \dots, n.$$

Set  $X^* = X^* + Y^p$ .

If  $\sum_{i=1}^m u_i^p = 0$  go to Step 3, otherwise set  $b = U^p$  and go to Step 1.

Step 3: The vector  $X^*$  is the heuristic optimum solution of the given CSS problem. STOP.



Observation.

It has been computationally observed that deleting all those constraints for which  $b_1$  is zero while solving the LP in Step 1 results in significant reduction in computational time.

2.4 Numerical Example:

We now solve a numerical example to illustrate various steps involved in the LP-based heuristic algorithm. Consider the following hypothetical CSS problem.

$$\text{Min. } \sum_{j=1}^{12} x_j$$

s.t.

1 0 0 0 0 1 0 1 1 0 1 1	$x_1$	123
1 1 0 0 0 0 1 0 1 1 0 1	$x_2$	123
1 1 1 0 0 0 0 1 0 1 1 0	$x_3$	122
0 1 1 1 0 0 0 0 1 0 1 1	$x_4$	125
1 0 1 1 1 0 0 0 0 1 0 1	$x_5$	123
1 1 0 1 1 1 0 0 0 0 1 0	$x_6$	124
0 1 1 0 1 1 1 0 0 0 0 1	$x_7$	124
1 0 1 1 0 1 1 1 0 0 0 0	$x_8$	122
0 1 0 1 1 0 1 1 1 0 0 0	$x_9$	123
0 0 1 0 1 1 0 1 1 1 0 0	$x_{10}$	124
0 0 0 1 0 1 1 0 1 1 1 0	$x_{11}$	124
0 0 0 0 1 0 1 1 0 1 1 1	$x_{12}$	123

$$x_j \geq 0, \text{ integer, } \forall j = 1, \dots, 12.$$

Step 1: This problem is first solved by relaxing the integrality restriction on all  $x_j$ 's as an ordinary LP. The optimal continuous valued variable vector is:

$$x = (39.80, 0.80, 40.60, 0.80, 41.40, 0.40, 40.40, 0.00, 41.60, 0.00, 40.80, 0.40)$$

Step 2: The  $n$  rounded-off solutions are:

$$\begin{aligned} y^1 &= (40, 1, 41, 0, 42, 0, 41, 0, 41, 0, 41, 0) \\ y^2 &= (39, 1, 41, 1, 41, 0, 41, 0, 41, 0, 41, 1) \\ y^3 &= (40, 0, 41, 1, 41, 1, 40, 0, 42, 0, 40, 1) \\ y^4 &= (40, 1, 40, 1, 42, 0, 40, 0, 42, 0, 41, 0) \\ y^5 &= (40, 1, 41, 0, 42, 0, 41, 0, 41, 0, 41, 0) \\ y^6 &= (40, 1, 40, 1, 41, 1, 40, 0, 42, 0, 41, 0) \\ y^7 &= (39, 1, 41, 1, 41, 0, 41, 0, 41, 0, 41, 1) \\ y^8 &= (40, 1, 40, 1, 42, 0, 40, 0, 42, 0, 41, 0) \\ y^9 &= (40, 1, 40, 1, 42, 0, 40, 0, 42, 0, 41, 0) \\ y^{10} &= (39, 1, 41, 1, 41, 0, 41, 0, 41, 0, 41, 1) \\ y^{11} &= (39, 1, 41, 1, 41, 0, 41, 0, 41, 0, 41, 1) \\ y^{12} &= (40, 0, 41, 1, 41, 1, 40, 0, 42, 0, 40, 1) \end{aligned}$$

The unsatisfied demand vectors and the total unsatisfied demand for each of the above 12 solutions are given in Table 3.1.

day	1	2	3	4	5	6	7	8	9	10	11	12	T.U.S. Demand
$U_1$	1	0	0	1	0	0	0	0	0	0	1	0	3
$U_2$	1	0	0	0	0	1	0	0	0	1	0	0	3
$U_3$	0	0	1	0	0	1	0	0	0	0	0	1	3
$U_4$	0	0	0	0	0	0	1	1	0	0	0	0	2
$U_5$	1	0	0	1	0	0	0	0	0	0	1	0	3
$U_6$	0	0	0	0	1	0	1	0	0	0	0	1	3
$U_7$	1	0	0	0	0	1	0	0	0	1	0	0	3
$U_8$	0	0	0	0	0	0	1	1	0	0	0	0	2
$U_9$	0	0	0	0	0	0	1	1	0	0	0	0	2
$U_{10}$	1	0	0	0	0	1	0	0	0	1	0	0	3
$U_{11}$	1	0	0	0	0	1	0	0	0	1	0	0	3
$U_{12}$	0	0	1	0	0	1	0	0	0	0	0	1	3

Table 3.1

We find that the total unsatisfied demand (T.U.S. demand) corresponding to the solution  $Y^4$  is minimum. Therefore,

$$X^* = (40, 1, 40, 1, 42, 0, 40, 0, 42, 0, 41, 0)$$

Step 3. The new demand vector for the CSS problem is

$$b = (0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0).$$

Solve the following LP, with this demand vector,

$$\text{Min. } \sum_{j=1}^{12} x_j$$

s.t.

$$x_2 + x_3 + x_5 + x_6 + x_7 + x_{12} > 1$$

$$x_1 + x_3 + x_4 + x_6 + x_7 + x_8 \geq 1$$

$$x_j \geq 0.$$

After solving this LP, we get,

$$X = (0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$

Since this is an integer solution, if we round it off in 12 ways, we get the same solution. The total unsatisfied demand is zero with this solution. Thus  $Y^P = X$ .

Therefore,

$$\begin{aligned} X^* &= X^* + Y^P \\ &= (40, 1, 41, 1, 42, 0, 40, 0, 42, 0, 41, 0) \end{aligned}$$

This is the final solution to the given CSS problem.

In this case, this is also an optimum solution.

## CHAPTER III

### EXACT ALGORITHMS

#### 3.1 Introduction.

In this chapter, we develop two exact algorithms for solving the CSS problem. In Section 3.3, we develop a branch and bound algorithm for solving the CSS problem optimally. This algorithm is an adaptation of the general integer programming branch and bound algorithm. Some of the important features of the proposed algorithm are: (1) Sensitivity analysis is made to impose lower or upper bound type of constraint on any variable to significantly reduce the storage and computational time requirements, (2) The LP-based heuristic algorithm developed in Chapter II is applied at every candidate problem to get a good feasible solution. We first explain in detail, different strategies that are followed in the algorithm. The flow-chart for the ~~the~~ proposed branch and bound algorithm is presented. A step-by-step procedure, to perform sensitivity analysis for imposing lower (upper) bound constraint on some variable, is then given. A numerical example is also solved. In Section 3.4, we develop a dual fractional cutting plane algorithm for solving the CSS problem. We introduce two types of cuts in this algorithm. One cut is based on the objective function value and the other one is Gomory's fractional cut.

### 3.2 Notations:

To simplify discussion in this chapter, we make use of the following additional symbols and notations.

$Z$	optimum objective function value of the continuous valued CSS problem.
$Z_*$	lower bound of the solution for the CSS problem.
UB	upper bound of the solution for the CSS problem.
$\bar{Z}$	incumbent objective function value.
$X$	incumbent solution.
$X^k$	continuous valued solution of candidate problem $P^k$ .
$Z^k$	optimum objective function value of the continuous valued candidate problem $P^k$ .
$Z_I^k$	objective function value of the integer feasible solution obtained when the LP-based heuristic algorithm is applied by taking $X^k$ as the first LP solution.
$X_I^k$	Integer feasible solution obtained when the LP-based heuristic is applied at candidate problem $P^k$ .
$L$	array containing lower bounds on variables $x_1$ , $x_1, 1 = 1, \dots, m+n$ .
$U$	array containing upper bounds on variables $x_1, 1 = 1, \dots, m+n$ .
$B$	$\{b_1, b_2, \dots, b_m\}$ set of all basic variables, where $b_i$ denotes the index of the basic variable corresponding to the $i$ -th row.

### 3.3 Branch and Bound Algorithm:

The basic idea in this method is to investigate all the feasible solutions by partitioning the solution set into subsets by a process, known as branching. A lower bound is calculated for the solutions with in each subset. If the lower bound is smaller than the objective function value of the incumbent solution and the subset with the lower bound is feasible, further branching is stopped from this subset. This is known as fathoming. After each partitioning, those subsets with a bound that exceeds that of a known feasible solution are excluded from further partitionings. This is known as pruning. The partitioning continues until a feasible solution is found such that its value is not greater than the bound for any subset. The algorithm terminates when all the subsets are either pruned or fathomed. We now describe various strategies of the branch and bound algorithm for the CSS problem considered in this thesis.

A branch and bound algorithm is characterized by its (i) bounding strategy, (ii) fathoming strategy, (iii) pruning strategy, (iv) branching strategy, and (v) search strategy. These strategies for the suggested branch and bound algorithm for the CSS problem are discussed in the following sub-sections.

#### 3.3.1 Bounding Strategy:

First, we solve the given CSS problem by relaxing the integrality restriction on all variables. The lower bound for

the given CSS problem is the smallest integer value greater than or equal to the continuous valued objective function.

$$LB = Z = \lceil Z \rceil.$$

The lower bound for candidate problem  $P^k = \lceil Z^k \rceil$ .

Initially we do not have any incumbent, therefore,

$$\bar{Z} = \infty.$$

### 3.3.2 Fathoming Strategy:

We fathom a candidate problem  $P^k$ , if  $Z_I^k$  satisfies the following condition.

$$\lceil Z^k \rceil = Z_I^k$$

### 3.3.3 Pruning Strategy:

We prune a candidate problem  $P^k$ , if either of the following conditions are satisfied

$$\lceil Z^k \rceil > \bar{Z}, \text{ or}$$

candidate problem  $P^k$  is infeasible.

### 3.3.4 Branching Strategy:

If a candidate problem  $P^k$  is neither fathomed nor pruned, it is used for subsequent branching. To perform branching at a candidate problem  $P^k$ , we can follow different strategies. The variable selected for branching should be such that it results in the enumeration of minimum number of nodes for finding the optimum solution. Many strategies were computationally tried by us.



The following strategy produced consistently better results than the other strategies.

As observed earlier in Section 2.2, by rounding-off the solution obtained at candidate problem  $P^k$ , using (2.2.1.1) to (2.2.1.n) we get  $n$  integer solutions. Let  $Y^p$  be the solution with minimum amount of total unsatisfied demand and  $U^p$  be the unsatisfied demand vector corresponding to this solution. If  $\sum_{i=1}^m u_i^p$  is zero we fathom this candidate problem otherwise we select a variable  $x_s$  for further branching. The strategy used for selecting  $x_s$  is explained below.

We have  $U^p$  available with us. Now, find out,  $F_j$ , the number of days on which shift  $j$  is active during the days for which demand is not satisfied. For each of the basic variables we compute,  $f_j = \lceil x_j \rceil - x_j$ . We select a variable  $x_s$  such that  $F_s \times f_s \geq F_j \times f_j, \forall j = 1, \dots, n$ . It is observed that selecting a variable such as  $x_s$  for branching brings in maximum amount of change in the lower bound in most of the cases. Now, we give a step by step procedure to identify branching variable  $x_s$ .

Step. 0: Set  $E_j = F_j = f_j = 0, \forall j = 1, \dots, n$

Step. 1: For all  $i = 1, \dots, m$  if  $(u_i^p > 0)$  then

for all  $j = 1$  to  $n$  such that  $a_{ij} = 1$ , set  $F_j = F_j + 1$ ,

for all  $j = 1$  to  $n$  such that  $j \in B$ , set  $f_j = \lceil x_j \rceil - x_j$ .

For all  $j = 1$  to  $n$  such that  $j \in B$ , set  $E_j = F_j \times f_j$ .

Find out a variable  $x_s$  such that

$$E_s = \max_{1 \leq j \leq n} \{E_j\}$$

Select the variable  $x_s$  as the source variable for further branching.

It is computationally observed that imposing upper bound type of constraint first results in the enumeration of smaller number of nodes for most of the problems. Hence, this strategy has been implemented.

### 3.3.5 Search Strategy:

As far as searching strategy is concerned, depth first search is employed because of the ease in implementation and lesser storage requirements.

We give the flow chart of the proposed branch and bound algorithm in Fig. 3.1.

### 3.3.6 Sensitivity Analysis:

In this section, we present a step by step procedure to perform sensitivity analysis by implicitly considering lower bound or upper bound type of constraint on some variable  $x_s$ . Let  $B$  denote the set of all basic variables at any stage of the problem. Corresponding to this basis, the relative cost coefficients of the variables  $x_j$ ,  $j = 1, \dots, (m+n)$  are  $\bar{c}_j = c_j - \sum_{i=1}^n \bar{a}_{ij}$ . Set  $G = \{j: \bar{c}_j \geq 0\}$  and  $H = \{j: \bar{c}_j < 0\}$  where  $G$  and  $H$  respectively denote the variables at their lower bound and at their upper bounds

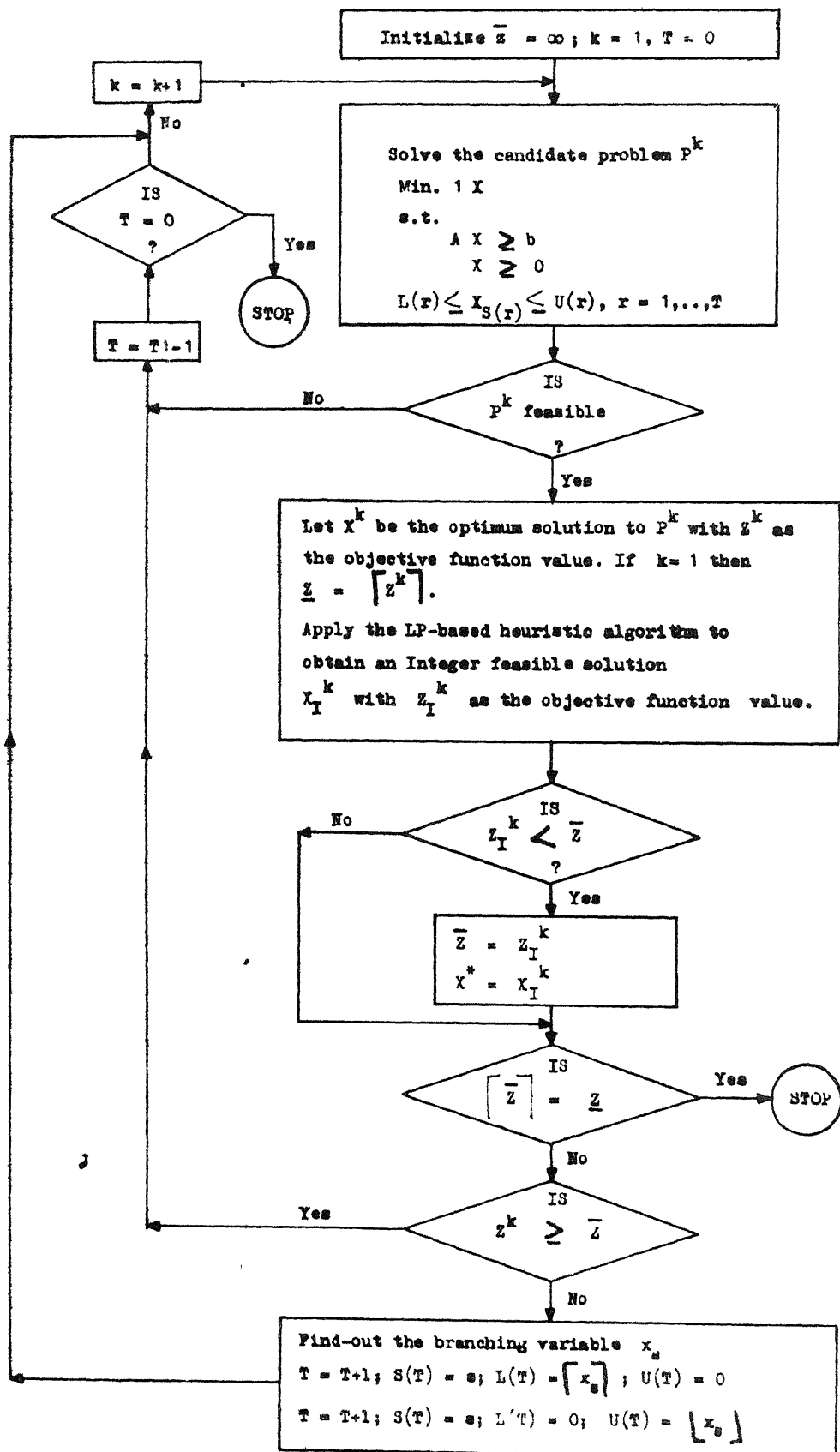


FIG. 3.1

Let  $(B, G, H)$  be an optimal basis structure of the CSS problem. Let  $A = \{\bar{a}_{1j}\} = B^{-1} A$ . Further, let  $\xi$  be the amount of change required in the current value of the variables  $x_s$  to satisfy the bound imposed on it. We first try to make the variable  $x_s$  non-basic, if it is a basic variable. If  $x_s$  is a basic variable and no variable is found to be eligible to enter the basis, the problem is infeasible. If  $s \in B$  and there is some variable  $t$  eligible to enter the basis, we make the variable  $x_s$  non-basic by performing one dual simplex iteration [20]. Now, we find out the maximum amount of required change that can be brought in the value of the variable  $x_s$  without changing the present basis structure. We denote this possible change by  $\lambda$ . If  $\lambda$  is greater than the required change  $\xi$ , we bring-in  $\xi$  amount of change in the value of the variable  $x_s$  and update the  $b$  column for the corresponding change and stop. If

$\lambda$  is less than  $\xi$ , only  $\lambda$  amount of change can be made in the value of the variable  $x_s$  without changing the present basis. We bring  $\lambda$  amount of change in the value of  $x_s$  and update  $\bar{b}$  column for the corresponding change brought in  $x_s$ . In this case, the bound restriction on the variable  $x_s$  is not yet satisfied because one of the basic variables reaches either its lower bound or its upper bound before the required change in the value of  $x_s$  can be brought in. The dual simplex iteration is again performed to replace the basic variable, which has reached its lower or upper bound recently, by an appropriate non-basic variable. The

basis structure is updated and this process is repeated until either  $\delta = 0$  or it is found that the dual simplex iteration can not be performed indicating infeasibility of the problem. A formal description of the procedure to perform sensitivity analysis is given below.

We consider the case when some lower bound restriction is imposed on variable  $x_s$ . We give the necessary changes required to be done if upper bound type of restriction is to be imposed on  $x_s$ .

Step 0: Let  $B$  denote the set of all basic variables,

$$G = \{j: \bar{c}_j \geq 0\}, \text{ and } H = \{j: \bar{c}_j < 0\}, \text{ where,}$$

$$\bar{c}_j = c_j - \sum_{i=1}^m a_{ij}, \quad \forall j = 1, \dots, n. \text{ Let } L \text{ and } U \text{ be two arrays respectively containing the tightest lower bound and upper bound restrictions on each variable } j,$$

$$\forall j = 1, \dots, m+n.$$

$$\text{Set } L(s) = x_s.$$

If  $s \notin B$  go to Step 2.

Let  $r$  be the row corresponding to  $x_s$  in  $\bar{A}$ .

Step 1: The variable  $x_{br}$  leaves the basis at its lower bound or upper bound. If  $x_{br}$  is equal to  $L(br)$  go to 1(a), else go to 1(b).

1(a): Let  $K = \{j: j \in G \text{ and } \bar{a}_{rj} < 0, \text{ or } j \in H \text{ and } \bar{a}_{rj} > 0\}$ . If  $K$  is empty, go to Step 3, otherwise identify a non-basic variable  $x_t$  for which

$-\bar{c}_t/\bar{a}_{rt} = \min_{j \in K} \{-\bar{c}_j/\bar{a}_{rj}\}$ , and go to 1(c).

1(b): Let  $K = \{j: j \in G \text{ and } \bar{a}_{rj} > 0, \text{ or } j \in H \text{ and } \bar{a}_{rj} < 0\}$ . If  $K$  is empty, go to Step 3, otherwise identify a non-basic variable  $x_t$  for which

$\bar{c}_t/\bar{a}_{rt} = \min_{j \in K} \{\bar{c}_j/\bar{a}_{rj}\}$ , and go to 1(c).

1(c): The variable  $x_{br}$  leaves the basis and  $x_t$  enters the basis. Update the  $\bar{A}$  matrix (except the column corresponding to  $\bar{b}$ ) by performing one dual simplex iteration. Go to Step 2.

Step 2: Update all  $x_j \in B$  as follows:

Compute  $\delta = \lceil \bar{L}(s) \rceil - L(s)$

If  $\delta$  is zero go to Step 4, otherwise compute,

$$\lambda_1 = \begin{cases} (\bar{b}_1 - L(b_1))/\bar{a}_{1s}, & \text{if } \bar{a}_{1s} > 0 \\ \infty, & \text{if } \bar{a}_{1s} = 0 \\ -(U(b_1) - \bar{b}_1)/\bar{a}_{1s}, & \text{if } \bar{a}_{1s} < 0 \end{cases}$$

Let  $\lambda = \min_{1 \leq i \leq m} \{\lambda_1, \delta\}$

Now update  $x_{bi} = x_{bi} - \bar{a}_{is} \cdot \lambda$

$$x_s = x_s + \lambda$$

$$L(s) = L(s) + \lambda$$

If  $\delta = \lambda$  go to Step 4 otherwise identify the row  $r$  corresponding to the minimum  $\lambda_i$  and go to Step 1.

Step 3: The problem is infeasible. STOP.

Step 4: The solution  $X$  is optimum and STOP.

For performing sensitivity analysis to impose upper bound type of constraint on variable  $x_s$ , we make the following changes in Step 0 and Step 2.

Step 0: Set  $U(s) = x_s$  instead of  $L(s) = x_s$ .

Step 2: Update all  $x_j \in B$  as follows.

Compute  $\delta = U(s) - \lfloor U(s) \rfloor$

If  $\delta$  is zero go to Step 4, otherwise,  
compute,

$$\lambda_1 = \begin{cases} (u(b_1) - \bar{b}_1)/\bar{a}_{1s}, & \text{if } \bar{a}_{1s} > 0 \\ \infty, & \text{if } \bar{a}_{1s} = 0 \\ -(\bar{b}_1 - L(b_1))/\bar{a}_{1s}, & \text{if } \bar{a}_{1s} < 0 \end{cases}$$

Set  $\lambda = \min_{1 \leq i \leq m} \{\lambda_i, \delta\}$

Now update  $x_{b_1} = x_{b_1} + \bar{a}_{1s} \cdot \lambda$

$$x_s = x_s - \lambda$$

$$U(s) = U(s) - \lambda$$

If  $\delta = \lambda$  go to Step 4, otherwise identify the row  $r$  corresponding to the minimum  $\lambda_1$  and go to Step 1.

If a node in the branching tree is either fathomed or pruned we have to get back to its parent, for this standard bounded linear programming techniques [15] can be used.

The algorithm is coded in FORTRAN-IV on DEC-1090 computer system and the listings are enclosed at the end. Extensive computational investigation is done. The performance of the algorithm is quite encouraging and is discussed in Chapter IV. We now take up a numerical example to illustrate the method.

### 3.3.7 Numerical Example:

We take up the same example that is considered in Chapter II. The branching tree enumerated is given in Fig. 3.2. Nodes are numbered by the order in which they are enumerated. The branching variable and the bound imposed on it is shown at each node. The lower bound obtained at each node is written in a rectangle at that node. Initially, the given CSS problem is solved as an ordinary LP, the continuous valued objective function value obtained is  $Z^1 = 247.00$   $[\bar{Z}^1] = [\overline{247.00}] = 247$  is the lower bound for the CSS problem. When LP-based heuristic is applied at node 1, we get a feasible solution with  $Z_I^1 = 248$ . Therefore,  $\bar{Z} = 248$ . The algorithm terminated after enumerating 25 nodes. In this problem the initial solution obtained at node 1 itself is the optimum solution.

$$X = (40, 1, 41, 1, 42, 0, 40, 0, 42, 0, 41, 0).$$



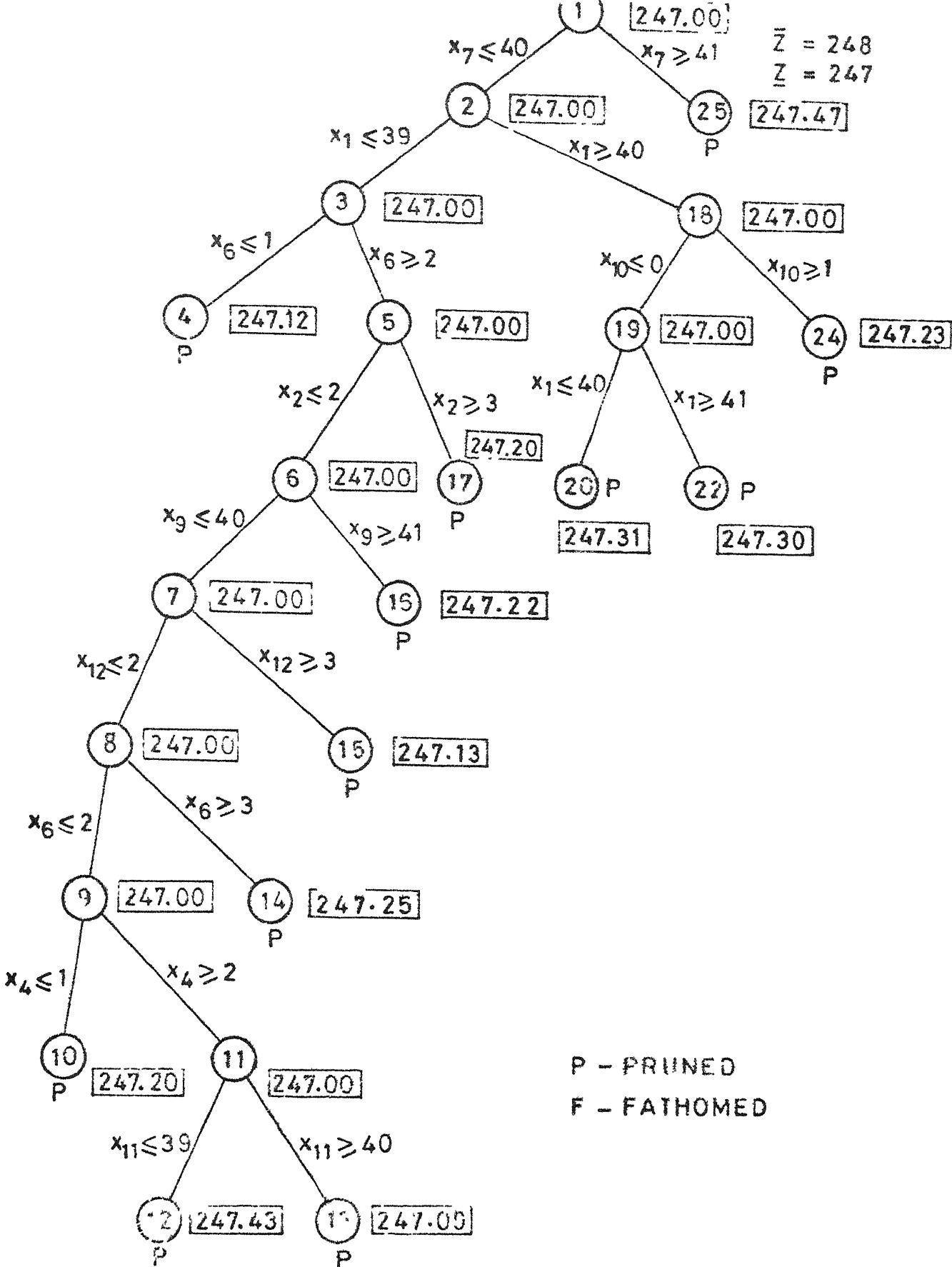


FIG. 3.2

### 3.4 Cutting Plane Algorithms

#### 3.4.1 Introduction

In this section, we develop a dual fractional cutting plane algorithm for solving the CSS problem. The general intent of cutting plane algorithms is to deduce supplementary **inequalities** from the integrality and constraint requirements and to append to the constraint set so as to eventually produce a linear program whose optimal solution is integer in the integer constrained variables. Gomory [17] developed the first cutting plane algorithm applicable to any Integer Program. Gomory [17] produced a second cutting plane algorithm for the integer program which requires only additions and subtractions in computation (an "all-integer" technique). All these methods maintain linear programs which are dual feasible, and are therefore often classified as dual cutting plane algorithms.

Glover and Young developed cutting plane algorithms for the Integer Program which maintain linear programs that are primal feasible. Since primal feasible integer solutions are successively produced, the technique is referred to as a primal cutting plane algorithm.

In this dissertation, we consider a cutting plane algorithm for the CSS problem which utilizes the lexicographic dual simplex method and allows fractional numbers in computation - hence the dual fractional reference. We take up the dual

fractional cutting plane algorithm because of its faster convergence rate.

### 3.4.2 Development of the Algorithm:

To avoid cycling and to ensure convergence we make use of lexicographic dual simplex method in the cutting plane algorithm.

Consider the problem,

$$\text{Max.} \quad C X = Z$$

s.t.

$$A X \leq b, \quad X \geq 0, \text{ integer}$$

This problem is represented in the following way in an inverse tableau to make use of lexicographic dual simplex method.

	1	$-x_1$	...	$-x_j$	....	$-x_n$	
$x_0 =$	$a_{00}$	$a_{01}$	...	$a_{0j}$	....	$a_{0n}$	
$x_1 =$	0	-1	...	0	....	0	$\geq 0$
:							:
$x_j =$	0	0	...	-1	....	0	$\geq 0$
:							:
$x_n =$	0	0	...	0	....	-1	$\geq 0$
$x_{n+1} =$	$a_{n+1,0}$	$a_{n+1,1}$	...	$a_{n+1,j}$	....	$a_{n+1,n}$	$\geq 0$
:							:
$x_{n+m} =$	$a_{n+m,0}$	$a_{n+m,1}$	...	$a_{n+m,j}$	....	$a_{n+m,n}$	$\geq 0$

where

$$x_0 = Z,$$

$$a_{00} = 0, \quad a_{0j} = -c_j, \quad \forall j = 1, \dots, n$$

$$a_{n+1,0} = b_1, \quad \forall 1 = 1, \dots, m$$

$$a_{n+1,j} = a_{1j}, \quad \forall 1 = 1, \dots, m, \quad \forall j = 1, \dots, n$$

In addition, we use the following notation:

$J$  contains the  $n$  indices of the current non-basic variables as they appear from left to right on the top most line of the tableau. Initially,  $J = \{1, 2, \dots, n\}$ .

$J(j)$  ( $j = 1, \dots, n$ ) is the  $j$ -th element in  $J$ .

Now, we explain how our cutting plane algorithm proceeds. We first solve the given CSS problem as an LP using lexicographic dual simplex method [15] and then round-off the continuous valued solution obtained in  $n$  ways using (2.2.1.1) to (2.2.1.n) to obtain  $Y^1, \dots, Y^n$ . Let  $Y^p$  be the solution satisfying the following inequality:

$$\sum_{i=1}^m u_i^p \leq \sum_{i=1}^m u_i^j, \quad \forall j = 1, \dots, n.$$

If  $U^p = 0$ , then  $Y^p$  is an optimum solution to the CSS problem and we stop. Otherwise, we impose a cut which eliminates the current non-integer solution and is satisfied by all the integer feasible solutions to the given CSS problem. We propose to impose two types of cuts. One cut is based on the

objective function value ( $\bar{Z}$ ) of the optimum solution and the other cut is Gomory's fractional cut.

Since the CSS problem is a pure integer program, the objective function value of the optimum solution to this problem must be an integer. Therefore, whenever we observe  $\lceil \bar{Z} \rceil > \bar{Z}$ , we append the following cut to inverse tableau.

$$\sum_{k=1}^n x_k \geq \lceil \bar{Z} \rceil \quad (3.4.2.1)$$

However, this cut can not be added directly in the above form. This problem can be tackled in the following way. Initially, before solving the LP, we append one more additional constraint  $\sum_{j=1}^n x_j \geq 0$  in the  $(m+n+1)$ -th row and then solve the LP. Let  $x_{m+n+1}$  be the slack variable corresponding to this constraint. We show that (3.4.2.1) is equivalent to,

$$x_{m+n+1} = \bar{Z} - \lceil \bar{Z} \rceil + \sum_{j=1}^n a_{m+n+1,j} (-x_J(j)) \quad (3.4.2.2)$$

The row associated with  $x_v$  in the current inverse tableau leads to constraint,

$$x_v = \bar{b}_v + \sum_{j=1}^n \bar{a}_{vj} (-x_J(j)) \quad (3.4.2.3)$$

Therefore, from (3.4.2.1) and (3.4.2.3)

$$\sum_{k=1}^n \bar{b}_k + \sum_{j=1}^n \left( \sum_{k=1}^n \bar{a}_{kj} \right) (-x_J(j)) \geq \lceil \bar{Z} \rceil$$

or,

$$\sum_{k=1}^n \bar{b}_k + \sum_{j=1}^n \left( \sum_{k=1}^n \bar{a}_{kj} \right) (-x_{J(j)}) \geq [\bar{Z}]$$

therefore,

$$\bar{Z} + \sum_{j=1}^n a_{m+n+1,j} (-x_{J(j)}) - x_{m+n+1} = [\bar{Z}]$$

$$x_{m+n+1} = \bar{Z} - [\bar{Z}] + \sum_{j=1}^n a_{m+n+1,j} (-x_{J(j)}).$$

Therefore, whenever we have to impose a constraint like (3.4.2.1), we simply make  $\bar{a}_{m+n+1,0} = (\bar{Z} - [\bar{Z}])$ . Now  $(m+n+1)$ -th constraint violates primal feasibility because  $(\bar{Z} - [\bar{Z}])$  is always negative. Apply dual simplex method to make the tableau primal feasible.

The second type of cut we impose is the standard Gomory's fractional cut. We impose this cut when  $\bar{Z}$  is an integer, but none of the  $n$  rounded-off solutions are feasible to the given CSS problem. If  $x_v$  is the source variable the following constraint is appended at the bottom of the inverse tableau.

$$x_{(m+n+1)+k} = -f_{v0} + \sum_{j=1}^n (-f_{vj}) (-x_{J(j)}) \geq 0 \quad (3.4.2.4)$$

where,  $x_{(m+n+1)+k}$  is Gomory's slack variable associated with  $k$ -th fractional cut added.

$$F_{vj} = a_{vj} - [a_{vj}], \quad \forall j = 0, 1, \dots, n$$

### 3.4.3 Source Variable Selection:

Proper selection of source variable for cut generation plays very important role in getting a fast optimal solution.

Let  $x_v$  be the source row. The hyperplane defining the cut is,

$$\sum_{j=1}^n f_{ij} x_{j(j)} = f_0.$$

We may say the larger the value of the ratio  $f_0/f_{ij}$  the stronger the cut is [17]. Hence, the objective when selecting a source row is to produce an inequality or hyperplane in which these ratios are as large as possible. Since source rows will usually yield inequalities in which the  $f_0/f_{ij}$  ratios are large for some variables and small for others, it is not clear as to which row to select. However, we intuitively feel that the following strategy produces a deep cut.

Step 1: Set  $f_i = \bar{b}_i - [b_i]$ ,  $\forall i = 1, \dots, m-n$  and  $F_1 = 0$ .

Arrange the  $f_i$ 's in decreasing order of their magnitude and select the first  $t$   $f_i$ 's.

Step 2: For each of the  $t$   $f_i$ 's selected in Step 1

Compute  $\sum_{j=1}^n f_{ij}$  and  $N_i$ , the total number of non-zero  $f_{ij}$ 's in the  $i$ -th row. Set  $F_1 = \sum \frac{f_i}{f_{ij}} N_i$

Let  $F_v = \max_{1 \leq i \leq m-n} \{F_i\}$

Select the variable  $x_v$  as the source variable for cut generation.

We have computationally observed that if the value of  $t$  is between 10 and 20, it would give better results. A formal statement of the algorithm is presented in the next section.

### 3.4.4 Statement of the Algorithm:

Step 1: Solve the following LP using Lexico-dual simplex method after transforming the given CSS problem into the inverse tableau form.

$$\text{Min. } 1^T X$$

s.t.

$$A X \geq b$$

$$\sum_{j=1}^n X_j \geq 0$$

$$X \geq 0$$

Let  $\bar{X}$  be the optimum solution obtained.

Step 2: Use  $n$  round-offs (2.2.1.1) to (2.2.1.n) to get  $y^1, \dots, y^n$ . Select  $y^p$  satisfying the following inequality.

$$\sum_{i=1}^m u_i^p \leq \sum_{i=1}^m u_i^j, \quad \forall j = 1, \dots, n$$

If  $U^p = 0$ , go to Step 5, otherwise go to Step 3.

Step 3: If  $[\bar{Z}] = \bar{Z}$  go to Step 4. Otherwise, introduce the cut (3.4.2.2). Perform necessary dual simplex iterations to get primal feasibility and go to Step 2.

Step 4: Find out the source row variable  $x_v$  using the routine described above and append the cut (3.4.2.3) to the bottom of the inverse tableau. Perform the dual simplex



iterations to make the problem primal feasible.

Go to Step 2.

Step 5:  $Y^P$  is the optimum solution to the CSS problem. STOP.

The computational performance of the algorithm is discussed in Chapter IV.

## CHAPTER IV

### COMPUTATIONAL INVESTIGATIONS

#### 4.1 Introduction:

In this chapter, we present the computational performance of LP-based heuristic algorithm developed in Chapter II, the branch and bound algorithm and the dual fractional cutting plane algorithm developed in Chapter III and Bartholdi's guaranteed accuracy round-off algorithm.

The computational experiments require a large number of problems. The details of the problems considered for computational study are given in Section 4.2. In Section 4.3, we discuss the various considerations such as accuracy, CPU time for analysing the three algorithms developed by us and Bartholdi's algorithm. Finally, in Section 4.4, the concluding remarks are given.

#### 4.2 Problems Considered for Experimentation:

We have considered five different sizes of problems for the computational study. The five sizes are (12 x 12), (24 x 24), (48 x 48), (72 x 72), and (96 x 96). To explain the significance of the problem size, for example, let us consider the problem of (24 x 24) size. These numbers indicate

that the planning horizon for scheduling is 24 time periods and there are 24 different types of shifts available for assigning the workers. The motivation behind considering these particular problem sizes is as follows. Normally, most of the service organizations such as telephone exchanges, hospitals, police and fire departments work round the clock. Depending on the demand, they can have a different shift starting every two hours, one hour, half an hour or quarter hour in a day. Thus, these particular sizes are considered.

As we have already discussed in Chapter I, a shift is differentiated from the other shifts by its start time or by its length or by the number of breaks or by the timing of breaks or by the combination of some or all of these. For a given problem, we differentiate two shifts by their start time in the problems considered. The cycle length, number of blocks of consecutive ones,  $q$ , and the timing of breaks are similar for all the  $n$  shifts.

Within each size of the problem, we considered the shifts with different values of  $q$ . The values of  $q$  considered are 2, 4, 6, and 8. For a given  $q$  and cycle length, we again considered two cases. In the first case, the cycle is symmetric about its middle elements and in the other case the cycle is non-symmetric. Altogether, 31 problems, 16 symmetric and 15 non-symmetric problems are considered. Tables 4.1 and 4.2 respectively provide the details of the symmetric and non-symmetric problems considered.

A cycle is an  $(m \times 1)$  column vector. Given a cycle, we can generate the A matrix by taking the  $(m \times 1)$  column vector and shifting it cyclically one position downward to obtain successive columns. Thus, the A matrix has the form

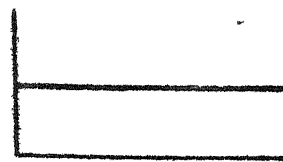
$$\begin{array}{cccc} a_1 & a_m & \dots & a_2 \\ a_2 & a_1 & \dots & a_3 \\ \vdots & \vdots & & \vdots \\ a_m & a_{m-1} & \dots & a_1 \end{array}$$

and is generated by the column vector  $(a_1, \dots, a_m)$ .

It is possible that not all the  $n$  shifts may be present in practice. As the number shifts becomes smaller, the problem becomes easier to solve. For all the problems are considered, all the  $n$  shifts are assumed to be present.

The demands during different time intervals were generated randomly by taking the range as 100 to 150.

In reality, there may be some pattern in the demand. To see how the performance of our algorithms is effected when there is certain pattern in demand, we have considered the following five demand patterns and solved two problems of  $(48 \times 48)$  size and  $(72 \times 72)$  size with  $q = 8$ . The demand patterns considered are as shown in Fig. 4.1.



LEVEL



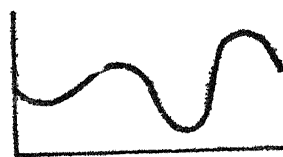
SLOPE



SINUSOIDAL



STEP



BIMODAL

Fig . 4.1 Various types of demand patterns.

Level: Here the demand is constant during all the time periods.

Slope: Here, the demand increases linearly from start time period to the end time period.

Step: In this case, the demand remains constant during first half of the time periods and then shoots up suddenly and remains at that value during rest of the time-periods.

Sinusoidal: In this case the demand pattern is like a sine wave.

Bimodal: Here, the demand pattern is like a sine wave during each half of the total time periods.

In Tables 4.1 and 4.2, we have given certain number for each problem. In the subsequent tables and discussion, these problems are referred to by their numbers.

We can observe that the CSS problem is completely specified, if we are given the problem size, cycle and the range of demands.

#### Computational Performance:

We, now give the computational performance of our algorithms and Bartholdi's algorithm in terms of accuracy and CPU times.

In Table 4.3, we give the objective function value obtained and the CPU times taken by Bartholdi's method, LP-based heuristic algorithm and the branch and bound algorithm for the symmetric problems. The total number of nodes enumerated in branch and bound algorithm is also given in the table.

It is evident from the table that the branch and bound algorithm was able to obtain optimum solution to all the problems. The maximum number of nodes enumerated are 53 for the problem 14. This problem has also taken the maximum amount of CPU time of 274.4 seconds.

The objective function values obtained using LP-based heuristic are observed to be very close to the optimum value in all the problems. The deviation from the optimum solution is less than 0.5 percent. The CPU time taken is much lesser when compared to branch and bound algorithm. This algorithm is observed to perform far better than Bartholdi's method in terms of accuracy especially when  $q$  is more.

It was observed that for none of the problems, the LP-based heuristic solved more than 2 LP's. In addition, the size of the second LP solved was often less than 10 percent of the original problem size.

Bartholdi's method was observed to be often taking as many additional units as given by his bound. It is better only in terms of the CPU time taken.

Similar data for the non-symmetric problems is given in Table 4.4.

The performance of branch and bound is somewhat discouraging in this case. 6 of the 15 problems were solved optimally by

enumerating less than 25 nodes. In those cases where the algorithm did not terminate after enumerating 100 nodes are marked with a star. For these problems we used  $\epsilon$  - approximation with  $\epsilon = 1$ . In this case, we terminate the problem if the incumbent solutions  $[\bar{Z}]$  satisfies the following condition:

$\bar{Z}$  is less than or equal to  $(Z + \epsilon)$ .

The LP-based heuristic and Bartholdi's method behaved in the same way as they did for the symmetric problems.

Problems 8 and 12 were solved for each of the five demand patterns described in the earlier section and the results are given in Table 4.6. In these cases, the LP-based heuristic algorithm performance is same as for the earlier problems. The branch and bound algorithm was able to solve problem 8 optimally for all the demand patterns. Problem 12 could not be solved when the demand pattern is constant or sinusoidal. When 1 unit of  $\epsilon$  - approximation is applied, these two cases were also solved. From the nodes enumerated, we can observe that the problem becomes more difficult to solve when there is certain pattern in demand.

The performance of the dual fractional cutting plane algorithm was quite erratic. The problems upto the size (72 x 72) were tried on this algorithm. Out of the 23 problems tried only 7 problems were solved and the others did not terminate after appending 150 cuts. The results of those problems



which were solved are given in Table 4.7. The algorithm could not solve problem sizes more than  $(48 \times 48)$ . In those cases, it could solve the problem, the number of cuts generated are very less in number.

#### 4.4 Concluding Remarks:

In Chapter I, we observed that the CSS problem with intermittently available resources is more difficult to solve. We gave the statement of the problem and reviewed the literature available for the CSS problem. We have developed three algorithms to solve the CSS problem more accurately.

The LP-based heuristic algorithm developed in Chapter II makes use of an improved version of the rounding-off procedure suggested by Bartholdi et al [5] and solves a sequence of LP's to get a good feasible solution to the CSS problem.

We developed a branch and bound algorithm in Chapter III. The salient features of this algorithm are: (1) It makes use of the LP-based heuristic algorithm at every node to get a good feasible solution, (2) Sensitivity analysis was used to significantly reduce the storage and computational time requirements, and (3) the criterion used for source row selection makes pruning more effective. We have also developed a dual fractional cutting plane algorithm in Chapter III. We proposed two types of cuts, one cut is based on objective function value and the other cut is Gomory's fractional cut.

Finally, we have done extensive computational study on all these three algorithms and Bartholdi's algorithm. We observed that the performance of the LP-based heuristic algorithm was quite encouraging. The deviation of the solution obtained using this algorithm from the optimum solution was not more than 2 units generally. We observed that not more than 2 LP's were needed to be solved for any problem in the LP-heuristic algorithm and the size of the second LP solved is often less than 10 percent of the original problem size. The performance of the heuristic was invariant with the pattern of demand.

The performance of the branch and bound algorithm is quite encouraging. Symmetric problems of the size  $(96 \times 96)$  with  $q = 8$  were solved optimally in a few minutes of CPU time. We have observed that problem becomes more difficult to solve when the cycle is non-symmetric or there is certain pattern in demand.

The performance of cutting plane algorithm was observed to be erratic.

We have considered the CSS problem when all  $c_j$ 's are 1. Further scope for research in this area would be to extend the ideas developed in this thesis to consider the case when  $c_j$ 's are different.

Table 4.1: Symmetric Problems.

Problem Number	Size MxN	q	Cycle Length	k	Speed	Cycle
1	12 x 12	4	7	4	10	1 0 1 0 1 0 1
2	24 x 24	2	9	8	10	1 1 1 1 0 1 1 1 1 1
3	24 x 24	4	11	8	30	1 1 0 1 1 0 1 1 0 1 1
4	24 x 24	6	13	8	50	1 1 0 1 0 1 0 1 0 1 0 1 1
5	48 x 48	2	18	16	10	1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1
6	48 x 48	4	20	16	30	1 1 1 1 0 1 1 1 1 1 0 0 1 1 1 1 0 1 1 1 1
7	48 x 48	6	22	16	55	1 1 1 0 1 1 1 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1
8	48 x 48	8	24	16	70	1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1
9	72 x 72	2	27	24	10	1 1
10	72 x 72	4	29	24	30	1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1
11	72 x 72	6	31	24	50	1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 0 0 1 1 1 0 1 1 1
12	72 x 72	8	33	24	70	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0 0 1 1 1 0 0 1 1 1 0 1 1 1 1
13	96 x 96	2	34	30	10	1 1
14	96 x 96	4	36	30	33	1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 0
15	96 x 96	6	38	30	50	1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 1 1
16	96 x 96	8	40	30	77	1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 0 0 1 1 1 0 0 0 0 1 1 1 0 1 1 1 1

Table 4.2: Non-Symmetric Problems.

Prob. No.	Size MxN	q	Cycle Length	K	Seed	Cycle
17.	24 x 24	2	9	8	20	1 1 1 1 1 0 1 1 1
18.	24 x 24	4	11	8	40	1 1 1 0 1 0 1 1 0 1 1
19.	24 x 24	6	13	8	60	1 0 1 1 0 1 0 1 1 0 1 0 1
20.	48 x 48	2	18	16	20	1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1
21.	48 x 48	4	20	16	40	1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 0 1 1 1
22.	48 x 48	6	22	16	60	1 1 1 1 0 1 1 0 1 1 1 0 0 1 1 0 1 1 1 0 1
23.	48 x 48	8	24	16	80	1 1 1 0 1 1 0 1 1 0 1 0 0 1 1 1 0 1 1 0 1
24.	72 x 72	2	27	24	20	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1
25.	72 x 72	4	29	24	40	1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1
26.	72 x 72	6	31	24	60	1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 0 0 0 1 1 1 1 0 1 1
27.	72 x 72	8	33	24	80	1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1
28.	96 x 96	2	34	30	20	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1
29.	96 x 96	4	36	30	40	1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 0 1 1
30.	96 x 96	6	38	30	60	1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 0 1 1 1 0
31.	96 x 96	8	40	30	80	1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 1 1 1

Table 4.3: Computational results for symmetric problems.

Prob. No.	Size MxN	Objective Function Value		CPU Time in Secs.		No. of nodes Examined in B and B
		Bartholdi's Method	Heuristic Method	Bartholdi's Method	Heuristic Method	
1.	12 x 12	408	399	0.1	< 0.1	1
2.	24 x 24	404	402	0.5	0.5	5
3.	24 x 24	409	400	0.2	0.3	1
4.	24 x 24	393	380	0.4	0.5	37
5.	48 x 48	395	393	3.2	3.7	18
6.	48 x 48	402	395	5.1	5.5	22
7.	48 x 48	394	380	3.6	3.9	26
8.	48 x 48	411	392	4.1	4.3	5
9.	72 x 72	400	398	11.8	13.4	2
10.	72 x 72	407	399	15.4	19.5	23
11.	72 x 72	416	404	19.3	19.6	9
12.	72 x 72	438	418	13.3	19.3	45
13.	96 x 96	438	436	35.7	35.9	4
14.	96 x 96	437	430	36.3	47.4	53
15.	96 x 96	445	431	38.7	49.2	9
16.	96 x 96	452	432	37.4	50.8	42

Table 4.4: Computational results for non-symmetric problems.

Prob. No.	Size MxN	Objective Function Value		CPU Time in Secs.		No. of nodes Examined in B and E.
		Bartholdi's Method	Heuristics Method	Bartholdi's Method	Heuristics Method	
17.	24 x 24	384	381	0.4	0.4	1
18.	24 x 24	392	384	0.4	0.4	1
19.	24 x 24	400	386	0.3	0.5	23
20.	48 x 48	393	391	2.9	3.5	20
21.	48 x 48	398	390	3.0	4.5	>17.1
22.	48 x 48	404	390	4.3	4.6	>103.1
23.	48 x 48	406	388	2.8	3.6	>118.5
24.	72 x 72	406	404	19.2	12.6	45.2
25.	72 x 72	399	391	17.7	19.5	>409.9
26.	72 x 72	414	402	18.4	19.1	>340.2
27.	72 x 72	420	401	21.5	24.6	>269.8
28.	96 x 96	444	441	30.7	28.7	29.3
29.	96 x 96	426	418	41.9	44.5	>698.8
30.	96 x 96	436	423	50.9	60.8	>804.8
31.	96 x 96	432	414	39.6	40.6	>911.7

Table 4.5: Performance of  $\epsilon$  - approximation method.

Prob. No.	Prob. Size	Objective Function Value B and B without $\epsilon$ -approximation	Objective Function Value B and B with $\epsilon$ -approximation	No. of Nodes Examined B and B without $\epsilon$ -approximation	No. of Nodes Examined B and B with $\epsilon$ -approximation	CPU Time B and B without $\epsilon$ -approximation	CPU Time B and B with $\epsilon$ -approximation
20	48 x 48	390	391	20	1	15.5	3.7
21	48 x 48	390	390	>100	1	117.1	4.6
22	48 x 48	390*	390	>100	1	103.1	4.8
23	48 x 48	386*	386	>100	70	118.5	76.9
24	72 x 72	403	404	19	1	45.2	13.1
25	72 x 72	391*	391	>100	1	409.9	20.1
26	72 x 72	400*	400	>100	9	340.2	38.5
27	72 x 72	400*	400	>100	10	269.8	41.3
28	96 x 96	441	441	1	1	29.2	29.4
29	96 x 96	418*	418	>100	1	698.8	45.4
30	96 x 96	422*	422	>100	19	804.8	45.4
31	96 x 96	412*	412	>100	48	911.7	303.6

Table 4.6: Performance of heuristic and branch and bound algorithms for various demand patterns.

Prob. No.	Prob. size	Demand Pattern	Objective Value	Function	CPU Time (Sec.)	Nodes examined	ε-Approximation Method
			LP-based heuristic	B and B algorithm	LP-based heuristic	in B and B algorithm	CPU time (Sec.) examined Nodes
8	48 x 48	LEVEL	300	300	3.3	3.3	1
		SINUSOIDAL	377	376	3.5	17.5	17
		BIMODAL	393	392	3.8	4.1	3
		SLOPE	390	389	3.8	23.8	27
		STEP	391	389	4.1	12.5	10
12	72 x 72	LEVEL	378	376*	14.7	>566.2	>100
		SINUSOIDAL	378	377*	14.7	>366.1	>100
		BIMODAL	426	423	25.4	250.3	75
		SLOPE	398	395	22.4	166.8	69
		STEP	386	384	17.3	151.9	54
							206.7
							103.3
							61
							31



Table 4.7: Performance of cutting plane algorithm.

Prob. No.	Size	No. of cuts on objective function	No. of fractional cuts	CPU time in sec.	Optimum objective function value
1.	12 x 12	0	0	< 0.1	399
2.	24 x 24	0	0	0.4	401
3.	24 x 24	0	0	0.3	400
5.	48 x 48	0	4	8.3	391
6.	48 x 48	0	2	6.7	393
17.	24 x 24	1	13	2.6	381
20.	48 x 48	1	3	11.9	390

## REFERENCES AND BIBLIOGRAPHY

1. Ahuja, R.K., "Minimax Linear Programming Problem," to appear in Operations Research Letters.
2. Arthur, Jeffrey, L. and Ravindran, A., "A Multiple Objective Nurse Scheduling Model," AIIE Trans. 13, 55-60, (1981).
3. Baker, K.R., "Scheduling a Full-Time Workforce to Meet Cyclic Staffing Requirements," Mgt. Sci. 20, 1561-1568 (1974).
4. Baker, K.R., "Scheduling Full-Time and Part Time Staff to Meet Cyclic Requirements." OR Quart (UK), 25, 65-76 (1974).
5. Bartholdi, J.J., "A Guaranteed Accuracy Round-Off Algorithm for Cyclic Scheduling and Set Covering," O.R. 29, 501-510, (1981).
6. Bartholdi, J.J., J.B. Orlin and H.D. Ratliff, "Cyclic Scheduling via Integers Programs with Circular Ones," O.R. 28, 1074-1085 (1980).
7. Dantzig, G.B., "A Comment on Eddies Traffic Delay at Toll Booths", OR 28, 1074-1085 (1954).
8. Glover, F., and J.R. Mulvey, "Network Related Scheduling Models for Problems with Quasi-Adjacency and Block Adjacency Structure," Working Paper HBS 76-3 (1976), Graduate School of Business Administration, Harvard University.
9. Henderson, W.B., "Mathematical Models for Telephone Operator Shift Scheduling," Ph.D. Dissertation, Kranner Graduate School of Industrial Administration, Purdue University (1974).
10. Henderson, W.B., and Berry, W.L., "Heuristic Methods for Telephone Operator Shift Scheduling: An Experimental Analysis", Mgt. Sci. 22, 1372-1380 (1976).
11. Laporte, G., and Moberg, Y., "Rotating Schedules," EJOM 4, 24-30 (1980).

12. Luce, B.J., "A Shift Scheduling Algorithm," Paper Presented at the ORSA National Meeting, San Diego (1973).
13. Mabert, V.A., and Watts, C.A., "A Simulation Analysis of Tour-Shift Construction Procedures," Mgt. Sci. 28, 520-532 (1982).
14. Morris, J.G., and Showalter, M.J., "Simple Approaches to Shift, Days-Off and Tour Scheduling Problem," Mgt. Sci. 29, 942-950 (1983).
15. Murty, K.G., "Linear and Combinatorial Programming," John Wiley and Sons, Inc., New York, 1976.
16. Oudhensden, D.L., and Wen-Jeng, Wu, "Telephone Operator Scheduling with a Fixed Number of Operators," EJOR 11, 55-59 (1982).
17. Salkin, H.M., "Integer Programming," Addison Wesley Publishing Company, Massachusetts, 1975.
18. Segal, M., "The Operator Scheduling Problem: A Network Flow Approach," OR 22, 808-823 (1974).
19. Shepardson, F., and R.E. Marsten, "A Lagrangean Relaxation Algorithm for the Two Duty Period Scheduling Problem," Unpublished Technical Report No. 532, The University of Arizona, Tucson, Arizona.
20. Willis, R.J., and Wilson, E.J.G., "Scheduling of Telephone Betting Operators - A Case Study," JORS 34, 991-998 (1983).
21. Wagner, H.M., "The Dual Simplex Algorithm for Bounded Variables," NRLQ 5, 257-262 (1958).

```

*****
LP-BASED HEURISTIC ALGORITHM
*****
INTEGER BVAR(100), IX(100), TIX(100), DEMAND(100), HIGH, CPUF
INTEGER ENTER, STPOSN(100), CL, AMOUNT, FAMUNT, TIX2(100), CYCLE(100)
REAL A(0:100, 0:200), RX(200), COST(100), GA(100, 100)
INTEGER SOLEM(100), USDEM(100), TUSDEM(100)
COMMON A, M, N
READ(23, *) NPROB
DO 120 I=1, NPROB
  WRITE(24, 10) I
  FORMAT(1X, 'PROBLEM - ', I2)
  WRITE(24, 20)
  FORMAT(5X, 'MULTIPLE LP APPROACH')
  READ(23, *) M, N, CL
  WRITE(24, 30) M, N
  FORMAT(5X, 'M = ', I3, ' N = ', I3)
  DO 40 J=1, M
    A(0, J)=0
    IF(J.LE.N) A(0, J)=1
  READ(23, *) ISEED, LOW, HIGH
  WRITE(24, 50) ISEED, LOW, HIGH
  FORMAT(5X, 'SEED = ', I3, ' RANGE OF DEMANDS IS ', I3, ' to ', I3)
  WRITE(24, 55) (CYCLE(I), I=1, CL)
  FORMAT(5X, 'CYCLE: ', 36I2)
  CALL GENRAT(ISEED, LOW, HIGH, CYCLE, CL, GA, DEMAND, STPOSN)
  READ(23, *) (DEMAND(I), I=1, *)
  DO 44 I=1, M
    A(1, I)=-DEMAND(I)
    A(0, I)=0.
    TC=0.
    MD=M
    DO 60 I=1, M
      DO 60 J=1, N
        A(1, J)=-GA(I, J)
      CALL RTIME(11)
      NA=M+M
      NB=0
      DO 70 I=1, N
        TIX(I)=0
        BIGM=100000
        EPS=0.0001
        NOLPS=0
        ITER=0
      80 NOLPS=NOLPS+1
      DO 100 J=N+1, NA
        DO 90 I=1, M
          A(1, J)=0
          IF((J-N).EQ.1) A(1, J)=1
          A(0, J)=0
        100 BVAR(J-N)=J
          A(0, NB)=0
          ITER=ITER+1
          BMIN=-EPS
          DO 120 I=1, M
            IF(A(1, NB).GE.BMIN) GO TO 120
            LEAVE=I
            BMIN=A(1, NB)
          120 CONTINUE
            IF(BMIN.EQ.-EPS) GO TO 170
            RMIN=BIGM
            DO 130 I=1, NA
              IF(A(LEAVE, I).GT.-EPS) GO TO 130
              RATIO=-A(0, I)/A(LEAVE, I)
              IF(RATIO.GE.RMIN) GO TO 130
              RMIN=RATIO
              ENTER=I
            130 CONTINUE
              BVAR(LEAVE)=ENTER
              FACTOR=A(LEAVE, ENTER)
              DO 140 J=0, NA
                A(LEAVE, J)=A(LEAVE, J)/FACTOR
              DO 150 I=0, M
                IF(I.EQ.LEAVE) GO TO 160
                A=A(1, ENTER)
              DO 150 J=0, NA
                A(1, J)=A(1, J)-A*A(LEAVE, J)
            150 CONTINUE
            GO TO 110
          160 CONTINUE
            IF(NOLPS.GT.1) GO TO 190
            XY=-A(0, NB)

```

```

179 30 STPDEM(I)=OTF*(I-1)+1
180 CALL SETRAN(ISEED)
181 DO 40 I=1,N
182 U1=KAN(X)
183 IDEM=LOW+(HIGH-LOW)*U1+0.5
184 DEMAND(I)=IDEM
185 A(I,0)=-IDEM
186 40 CONTINUE
187 RETURN
188 END

```

\*\*\*\*\*  
 BRANCH AND BOUND ALGORITHM FOR  
 CYCLIC STAFF SCHEDULING PROBLEM  
 \*\*\*\*\*

IX=array containing final solution.

-1 FOR VARIABLES AT THEIR LOWER BOUND

STATUS = 0 FOR BASIC VARIABLES

1 FOR VARIABLES AT THEIR UPPER BOUND

INTEGER I,VAR(100),DEMAND(100),ENTER,STATUS(200),TOP,SP,UBOBJ,OB

INTEGER STACK(500,3),IX(100),SV,HIGH,FEAS,BVAR1(100),CYCLE(100)

INTEGER GA(100,100),IX1(100),TC,STPOSH(100),SOE1(100),JSOEN(100)

1ADJUST,CL,TIX(100),USDEM1(100)

REAL A(101,200),RX(200),COST(100),UB(200),LB(200),AI(101,200)

COMMON A,BVAR,STATUS,UB,LB,NA,NB,MC,EPS,XLARGE,I,N,STACK,TOP,

IGA,STPOSH,SDEM,USDEM,CL,DEMAND

READ(21,\*)NPROB

DO 310 I=1,NPROB

WRITE(22,10)I

10 FORMAT(1X,'PROBLEM ',12)

READ(21,\*)I,N,CL

READ(21,\*)ISEED,LOW,HIGH

READ(21,\*)(CYCLE(I),I=1,CL)

15 WRITE(22,15)(CYCLE(I),I=1,CL)

FORMAT(5X,'CYCLE:',13612)

WRITE(22,20)I,N

20 FORMAT(5X,'I = ',14,' N = ',14)

WRITE(22,30)ISEED,LOW,HIGH

30 FORMAT(5X,'SEED = ',14,' DEMAND RANGE IS ',15,' to ',14)

IC=I+1

JA=I+1

IB=I+I+1

CALL GENRAT(ISEED,LOW,HIGH,CYCLE)

AI,IGA=0

DO 41 I=1,N

41 A(I,IB)=-DEMAND(I)

CALL RTIME(1)

DO 40 J=1,SP

A(4C,I)=0

40 IF(J.LE.1)A(MC,I)=1

DO 50 I=1,N

DO 50 J=1,N

50 A(I,J)=-GA(I,J)

50 CONTINUE

BIGB=1000000.

XLARGE=1000000.

EPS=.0001

IPER=0

DO 70 I=1,NA

STATUS(I)=-1

IF(I.GT.N)STATUS(I)=0

LB(I)=0.

70 JB(I)=XLARGE

DO 80 J=N+1,NA

DO 80 I=1,N

A(I,J)=0

80 IF((J-I).EQ.1)A(I,J)=1

CONTINUE

A(MC,J)=0

BVAR(J-N)=J

CONTINUE

A(MC,NB)=0

100 IPER=IPER+1

BMIN=-EPS

DO 110 I=1,N

IF(A(I,NB).GE.BMIN)GO TO 110

LEAVE=1

BMIN=A(I,NB)

110 CONTINUE

IF(BMIN.EQ.-EPS)GO TO 160

RAT1=BIGB

DO 120 J=1,NA

IF(A(LEAVE,J).GT.-EPS)GO TO 120

RATIO=-A(MC,J)/A(LEAVE,J)

IF(RATIO.GE.RMIN)GO TO 120

RMIN=RATIO

ENTER=J

120 CONTINUE

STATUS(BVAR(LEAVE))=-1

STATUS(ENTER)=0

BVAR(LEAVE)=ENTER

RATIO=A(LEAVE,ENTER)

DO 130 J=1,NB

```

130 A(LEAVE,J)=A(LEAVE,J)/FACTOR
DO 150 I=1,10
IF(I.EQ.LEAVE)GO TO 150
X=A(I,ENTER)
DO 140 J=1,10
140 A(I,J)=A(I,J)-X*A(LEAVE,J)
CONTINUE
GO TO 130
160 XY=-A(40,48)
WRITE(22,170)XY
170 FORMAT(5X,'FIRST LP SOLUTION = ',F10.3)
LBOBJ=XY+1.9999
FCI=J
TOP=J
KOUNT=1
UBOBJ=XLARGE
CALL FLOW(FCI,UBOBJ,TIX,AMOUNT)
C WRITE(22,180)(TIX(J),J=1,N)
181 FORMAT(2X,'TIX(1) = ',B14)
XY=-A(40,48)
WRITE(22,190)AMOUNT
191 FORMAT(2X,'AMOUNT = ',14)
IF(AMOUNT.EQ.0)GO TO 210
IF(TIX.EQ.UBOBJ)GO TO 200
UBOBJ=FCI
WRITE(22,193)UBOBJ
193 FORMAT(2X,'UBOBJ = ',I10)
DO 190 I=1,8
190 IX(I)=TIX(I)
200 IF((UBOBJ+ALPHA).GE.UBOBJ)GO TO 290
IF(XY.GT.(UBOBJ+EPS-1))GO TO 230
CALL SUBROUT(SV,SR)
C WRITE(22,210)SR,SV
211 FORMAT(2X,'SR = ',14,'SV = ',14)
TOP=TOP+1
STACK(TOP,1)=I
STACK(TOP,2)=SV
STACK(TOP,3)=A(SR,NB)
UBJ=STACK(TOP,3)
LEAVE=SR
WRITE(22,213)UBJ,SV,LEAVE
213 FORMAT(2X,'UBJ = ',14,'SV = ',14,' LEAVE = ',14)
CALL SUBROUT(LEAVE,UBJ,FEAS)
C WRITE(22,215)A(40,NB)
216 FORMAT(2X,'A(40,NB) = ',F10.3)
KOUNT=KOUNT+1
IF(KOUNT.GE.100)GO TO 290
IF(FEAS.EQ.0)GO TO 230
XY=-A(40,48)
WRITE(22,220),UBOBJ,UBOBJ,XY
C 220 FORMAT(1X,'UBOBJ = ',10,'LBOBJ = ',15,' OBJ = ',F9.3)
IF(XY.GT.(UBOBJ+EPS-1))GO TO 230
GO TO 180
230 CALL SUBROUT
XY=-A(40,48)
IF(XY.GT.(UBOBJ+EPS-1))GO TO 240
IF(STACK(TOP,1).EQ.1)GO TO 250
TOP=TOP-1
IF(TOP.EQ.0)GO TO 290
GO TO 230
250 SV=STACK(TOP,2)
STACK(TOP,1)=-1
STACK(TOP,3)=STACK(TOP,3)+1
DO 260 I=1,8
260 IF(SVAR(I).EQ.SV)GO TO 270
I=0
270 LEAVE=I
LBL=STACK(TOP,3)
CALL LBOUND(LEAVE,LBL,FEAS)
KOUNT=KOUNT+1
IF(KOUNT.GE.100)GO TO 290
IF(FEAS.EQ.0)GO TO 230
XY=-A(40,48)
WRITE(22,*)KOUNT,TOP
C WRITE(22,280)UBOBJ,LBOBJ,XY
280 FORMAT(1X,'UBOBJ = ',18,'LBOBJ = ',15,' OBJ = ',F9.3)
IF(XY.GT.(UBOBJ+EPS-1))GO TO 230
GO TO 180
290 CALL RTIME(N2)
TIME=TIME-RTIME
300 WRITE(22,300)IRTIME
FORMAT(5X,'CPU TIME = ',110,' milli seconds')
WRITE(22,310)UBOBJ
310 FORMAT(5X,'THE OBJ VALUE = ',18)
C WRITE(22,1320)(I,IX(I),I=1,N)
1320 FORMAT(5X,'X-',12,'=',14)
WRITE(22,320)KOUNT
320 FORMAT(5X,'NO. OF NODES EXAMINED = ',14)

```

1800  
1810  
1820  
1830

330  
340

FORMAT(///)  
CONTINUE  
STOP  
END



```

*****
THIS SUBROUTINE GENERATES THE DATA REQUIRED FOR
THE CYCLIC STAFF SCHEDULING PROBLEM
*****
SUBROUTINE GENERATE(ISEED,LOW,HIGH,CYCLE)
INTEGER CYCLE(100),DIF,0
INTEGER BVAR(100),DEMAND(100),ENTER,STATUS(200),TJP,SR,UBOBJ,OB
INTEGER SPACK(300,3),IX(100),SV,HIGH,FEAS,BVAPI(100)
INTEGER GA(100,100),IX1(100),TC,STPDSN(100),SDEM(100),USDEM(100)
IAMPJN1,CL,PLX(100),USDEM1(100)
REAL A(101,200),RX(200),COST(100),UB(200),LB(200),AI(101,200)
COMMON A,BVAR,STATUS,UB,LB,NA,NA,AC,EPS,XLARGE,M,N,STACK,TOP,
IGH,STPDSN,SDEM,USDEM,CL,DEMAND
DO 10 I=1,M
DO 10 J=1,N
GA(I,J)=0
DIF=M/N
DO 20 J=1,N
K=1
DO 20 I=1,CL
L=1+DIF*(J-1)
IF(L.GT.M) L=L-M
GA(L,J)=CYCLE(K)
20 K=K+1
DO 30 J=1,N
STPDSN(J)=DIF*(J-1)+1
CALL SETRAN(1SEED)
DO 40 I=1,M
U1=TRAN(X)
IDEM=LB+(HIGH-LB)*U1+0.5
OTRAN(I)=IDEM
A(I,NB)=-IDEM
40 CONTINUE
J=0
DO 50 J=1,N
L=J+1
IF(J.EQ.1) L=1
IF((GA(1,J).EQ.0).AND.(GA(1,L).EQ.1)) O=O+1
50 CONTINUE
K=0
DO 60 I=1,CL
IF(CYCLE(I).EQ.1) K=K+1
IF(E(22,70)) 2,K
70 FORMAT(5X,'O' = ',12,' K = ',12)
RETURN
END

```

```

*****
THE FOLLOWING SUBROUTINE ROUNDS-OFF THE FIRST LP
SOLUTION AND SOLVES SECOND LP IF NEEDED.
*****
SUBROUTINE INCUMB(TC1,URUBJ,TIX,AMOUNT)
INTEGER BVAP(100),DEMAND(100),ENTER,STATUS(200),TOP,SR,UBUBJ,OB
INTEGER STACK(300,3),IX(100),SV,HIGH,FEAS,BVARI(100)
INTEGER GA(100,100),IX1(100),TC,STPOS(100),SDEM(100),USDEM(100)
AMOUNT,CL,PIY(100),USDEM1(100)
READ 1(101,200),RX(200),COST(100),UB(200),LB(200),A1(101,200)
COMMON A,BVAR,STATUS,UB,LB,NA,NB,MC,EPS,XLARGE,M,N,STACK,TOP,
IGA,STPOS,SDEM,USDEM,CL,DEMAND
DO 30 I=1,M
TIX(I)=0
IF(STATUS(I))10,30,20
RX(I)=LB(I)
GO TO 30
20 RX(I)=UB(I)
CONTINUE
30 DO 40 I=1,M
RX(BVAR(I))=A(1,NB)
CALL ROUND(RX,IX1,TIX,TC,AMOUNT,USDEM1)
DO 50 I=1,N
TIX(I)=IX1(I)
TC1=TC
IF(AMOUNT.EQ.0)RETURN
DO 60 I=1,M
USDEM(I)=USDEM1(I)
IF((UBUBJ-TC1).LE.1)RETURN
M1=0
DO 80 I=1,M
IF(USDEM(I).LE.0)GO TO 80
M1=M1+1
A1(M1,NB)=-USDEM1(I)
DO 70 J=1,N
A1(M1,J)=-GA(I,J)
CONTINUE
DO 90 J=1,N
A1(M1+1,J)=1
A1(M1+1,NB)=0
CALL NEXTLP(M1,A1,RX,IX1,TIX,TC,AMOUNT,USDEM1)
IF(AMOUNT1.EQ.0)RETURN
DO 100 J=1,N
TIX(J)=TIX(J)+IX1(J)
TC1=TC1+TC
RETURN
END

```

```

000 ***** THIS SUBROUTINE SOLVES ONE LP *****
***** SUBROUTINE NEXLP(MI,AI,RX,IX,TIX,TC,AMOUNT,USDEM1) *****
INTEGER BVAR(100),DEMAND(100),ENTER,STATUS(200),TOP,SR,UBOBJ,OB
INTEGER STACK(300,3),IX(100),SV,HIGH,FEAS,BVARI(100)
INTEGER GA(100,100),IX1(100),TC,STPSN(100),SDEM(100),USDEN(100)
IAMOUNT,CCL,TLX(100),USDEM1(100)
REAL A(101,200),RX(200),COST(100),UB(200),LB(200),
IA(101,200)
COMMON A,BVAR,STATUS,OB,LB,UA,NB,MC,EPS,XLARGE,M,N,STACK,TOP,
IGA,STPSV,SDEM,USDEN,CCL,DEMAND
ITER=0
DO 20 J=N+1,N+M1
DO 10 I=1,M1
A1(I,J)=0
10 IF((J-N).EQ.1)A1(I,J)=1
A1(M1+1,J)=0
20 BVAR1(J-N)=J
A1(M1+1,NB)=0
30 ITER=ITER+1
BMIN=-EPS
DO 40 I=1,M1
IF(A1(I,NB).GE.BMIN)GO TO 40
LEAVE=I
BMIN=A1(I,NB)
CONTINUE
IF(BMIN.EQ.-EPS)GO TO 100
RMIN=XLARGE
DO 50 J=1,N+M1
IF(A1(LEAVE,J).GE.-EPS)GO TO 50
RATIO=-A1(M1+1,J)/A1(LEAVE,J)
IF(RATIO.GE.RMIN)GO TO 50
RMIN=RATIO
ENTER=J
CONTINUE
60 FORMAT(' RMIN = ',F12.2,' ENTER = ',I4)
BVAR1(LEAVE)=ENTER
FACTOR=A1(LEAVE,ENTER)
DO 70 J=1,NB
A1(LEAVE,I)=A1(LEAVE,J)/FACTOR
DO 90 I=1,M1+1
IF(I.EQ.LEAVE)GO TO 90
X=A1(I,ENTER)
DO 80 J=1,NB
A1(I,J)=A1(I,J)-X*A1(LEAVE,J)
CONTINUE
GO TO 30
DO 110 J=1,N
DO 120 I=1,M1
RX(J)=0
120 RX(BVAR1(I))=A1(I,NB)
CALL RJND(RX,IX1,TIX,IC,AMOUNT,USDEM1)
RETURN
END

```

```

*****
      THIS SUBROUTINE OF THE SENSITIVITY ANALYSIS TO IMPOSE
      UPPER BOUND CONSTRAINT ON ANY VARIABLE
*****
      SUBROUTINE HBOUND(LEAVE,UBL,FEAS)
      INTEGER ENTER,BVAR(100),STATUS(200),LEAVE,STACK(300,3),TOP,FEA
      INTEGER JA(100,100),IX1(100),TC,STPOSN(100),SDEN(100),USDEM(100
      1A0JNT,DEMAND(100),CL,TIX(100),USDEM(100)
      REAL A(101,200),UB(200),LB(200),LEMMIN,LEMDA
      COMMON A,BVAR,STATUS,UB,LB,NA,NB,NC,EPS,XLARGE,M,N,STACK,TOP,
      1GA,STPOSN,SDEN,USDEM,CL,DEMAND
      I1=STACK(TOP,2)
      LEAVE=LEAVE
      FEAS=1
      KOUNT=0
      UB(I1)=A(LEAVE,NB)
      XI1=A(LEAVE,NB)
      C=1
10    RMIN=XLARGE
      KOUNT=KOUNT+1
      DO 20 J=1,NA
      IF(LB(J).EQ.UB(J))GO TO 20
      XX=C*STATUS(J)*A(LEAVE,J)
      IF(XX.GE.-EPS)GO TO 20
      RATIO=C*A(NC,J)/A(LEAVE,J)
      IF(RATIO.GE.RMIN)GO TO 20
      RMIN=RATIO
      ENTER=J
20    CONTINUE
      IF(RMIN.NE.XLARGE)GO TO 50
      FEAS=0
      IF(KOUNT.GT.1)GO TO 40
      IF(LEAVE.EQ.0)GO TO 40
      UB(I1)=XLARGE
      IF(TOP.EQ.1)GO TO 40
      DO 30 J=1,(TOP-1)
      I=TOP-J
      IF(STACK(I,2).GE.I1)GO TO 30
      IF(STACK(I,1).EQ.-1)GO TO 30
      UB(I1)=STACK(I,3)
      GO TO 40
30    CONTINUE
40    RETURN
50    IL=BVAR(LEAVE)
      FACTOR=A(LEAVE,ENTER)
      DO 60 J=1,NA
      A(LEAVE,J)=A(LEAVE,J)/FACTOR
60    DO 80 I=1,NC
      IF(I.EQ.LEAVE)GO TO 80
      TEMP=A(I,ENTER)
      DO 70 J=1,NA
      A(I,J)=A(I,J)-A(LEAVE,J)*TEMP
70    CONTINUE
80    STATUS(IL)=-1
      IF(A(LEAVE,NB).GT.(LB(IL)+EPS))STATUS(IL)=1
      A(LEAVE,NB)=LB(ENTER)
      IF(STATUS(ENTER).EQ.1)A(LEAVE,NB)=UB(ENTER)
      BVAR(LEAVE)=ENTER
      STATUS(ENTER)=0
      LEMMIN=XLARGE
      DO 110 I=1,M
      LEMDA=XLARGE
      AC=A(I,I1)
      IF(ABS(AC).GE.EPS)GO TO 110
      IF(AC.GT.EPS)GO TO 90
      LEMDA=-(A(I,NB)-LB(BVAR(1)))/AC
      C1=-1
      GO TO 100
90    URI=UB(BVAR(1))
      IF(UB1.EQ.XLARGE)GO TO 100
      LEMDA=(UB1-A(I,NB))/AC
      C1=1
100   IF(LEMDA.GE.LEMMIN)GO TO 110
      LEMMIN=LEMDA
      LEMDA=N
      C=C1
110   CONTINUE
      CHANGE=XI1-UBL
      IF(LEMMIN.GT.CHANGE)LEMMIN=CHANGE
      DO 120 I=1,NC
      A(I,NB)=A(I,NB)+A(I,I1)*LEMMIN
120   XI1=XI1-LEMMIN
      UB(I1)=XI1
      IF(CHANGE.EQ.LEMMIN)GO TO 130
      LEAVE=LEMDA

```

001  
002  
003  
004

C

130

GO TO 10  
RETURN  
END

```

*****
      THIS SUBROUTINE DO THE SENSITIVITY ANALYSIS TO IMPOSE
      TO IMPOSE LOWER BOUND CONSTRAINT ON A VARIABLE
*****
      SUBROUTINE LBOUND(LEAVE,LBL,FEAS)
      INTEGER ENTER,BVAR(100),STATUS(200),JLEAVE,STACK(300,3),TOP,FEA
      READ A(101,200),UB(200),LB(200),LEMMIN,LEMDA
      INTEGER GA(100,100),IX1(100),PC,STPJSN(100),SDEM(100),USDEM(100)
      1A=JUNT,DEMAND(100),CL,TIX(100),USDEM(100)
      COMMON A,BVAR,STATUS,UB,LB,NA,NB,MC,EPS,XLARGE,M,N,STACK,TOP,
      LGA,STPJSN,SDEM,USDEM,CL,DEMAND
      IL=STACK(TOP,2)
      JLEAVE=LEAVE
      FEAS=1
      KOUNT=0
      IF(LEAVE.NE.0)GO TO 10
      XII=LB(IL);IF(STATUS(IL).EQ.1)XII=UB(IL)
      IF(XII.GE.LBL)RETURN
      STPJS(IL)=-1
      GO TO 100
10  LB(IL)=A(LEAVE,NB)
      XII=A(LEAVE,NB)
      C=-1
20  RMIN=XLARGE
      KOUNT=KOUNT+1
      DO 30 J=1,NA
      AX=C*STATUS(J)+A(LEAVE,J)
      IF(AX.GE.-EPS)GO TO 30
      IF(LB(J).GT.UB(J))GO TO 30
      RATIO=C+A(MC,J)/A(LEAVE,J)
      IF(RATIO.GE.RMIN)GO TO 30
      RMIN=RATIO
      ENTER=J
      CONTINUE
30  IF(RMIN.NE.XLARGE)GO TO 60
      FEAS=0
      IF(KOUNT.GT.1)GO TO 50
      IF(JLEAVE.EQ.0)GO TO 50
      LB(IL)=0
      IF(TOP.EQ.1)GO TO 50
      DO 40 J=1,(TOP-1)
      I=TOP-J
      IF(STACK(I,2).NE.I)GO TO 40
      IF(STACK(I,1).EQ.1)GO TO 40
      LB(IL)=STACK(I,3)
      GO TO 50
40  CONTINUE
50  RETURN
60  IL=BVAR(LEAVE)
      FACTOR=A(LEAVE,ENTER)
      DO 70 J=1,NA
      A(LEAVE,J)=A(LEAVE,J)/FACTOR
      DO 90 I=1,MC
      IF(I.EQ.LEAVE)GO TO 90
      TEMP=A(I,ENTER)
      DO 80 J=1,NA
      A(I,J)=A(I,J)-A(LEAVE,J)*TEMP
      CONTINUE
80  STPJS(IL)=-1
      IF(A(LEAVE,NB).GT.(UB(IL)+EPS))STATUS(IL)=1
      A(LEAVE,NB)=LB(ENTER)
      IF(STATUS(ENTER).EQ.1)A(LEAVE,NB)=UB(ENTER)
      BVAR(LEAVE)=ENTER
      STPJS(ENTER)=0
100  LEMMIN=XLARGE
      DO 130 I=1,M
      LEMDA=XLARGE
      AC=A(I,I)
      IF(ABS(AC).LE.EPS)GO TO 130
      IF(AC.LT.-EPS)GO TO 110
      LEMDA=(A(I,NB)-LB(BVAR(I)))/AC
      C1=-1
      GO TO 120
110  URI=JB(BVAR(I))
      IF(LBI.EQ.XLARGE)GO TO 120
      LEMDA=-(URI-A(I,NB))/AC
      C1=1
120  IF(LEMDA.GE.LEMMIN)GO TO 130
      LEMMIN=LEMDA
      LEMDA=1
      C1=C1
130  CHANGE=LBO-XII
      IF(LEMMIN.GT.CHANGE)LEMMIN=CHANGE

```

140

DO 140 I=1,MC  
A(I,NB)=A(I,NB)-A(I,11)\*LEMMIN  
X11=X11+LEMMIN  
UB(11)=X11  
IF(CHANGE.EQ.LEMMIN)GO TO 150  
LEAVE=UB(11)  
GO TO 20  
DO 150 I=1,MC  
CONTINUE  
RETURN  
END

150

160

```

C *****
C THIS SUBROUTINE DOES THE SENSITIVITY ANALYSIS TO RELAX
C A CONSTRAINT ON ANY VARIABLE.
C *****
SUBROUTINE GOSBACK
  INTEGER ENTER,BVAR(100),STATUS(200),STACK(300,3),TOP,STATIR,C1
  INTEGER SA(100,100),IX1(100),TC,STPDSN(100),SDEM(100),USDEM(100)
  IANDJNT,DEMAND(100),CL,FLAG,TIX(100),USDEM1(100)
  REAL A(101,200),UB(200),LB(200),LEMMIN,LEMDA
  COMMON A,BVAR,STATUS,UB,LP,NA,NB,MC,EPS,XLARGE,M,N,STACK,TOP,
  IGA,STPDSN,SDEM,USDEM,CL,DEMAND
  IR=STACK(TOP,2)
  STATIR=STATUS(IR)
  FLAG=1
  PVIR=LB(IR)
  IF(STATIR.EQ.1)PVIR=UB(IR)
  K2=STACK(TOP,1)
  IF((K2*STATIR).LE.0)FLAG=0
  IF(FLAG.EQ.1)GO TO 20
  IF(K2.EQ.1)UB(IR)=XLARGE
  IF(K2.EQ.-1)LB(IR)=0
  IF(TOP.EQ.1)RETURN
  DO 10 J=1,(TOP-1)
  I=TOP-J
  IF(STACK(I,2).NE.IR)GO TO 10
  IF(STACK(I,1).NE.K2)GO TO 10
  IF(K2.EQ.-1)LB(IR)=STACK(I,3)
  IF(K2.EQ.1)UB(IR)=STACK(I,3)
  RETURN
10 CONTINUE
  RETURN
20 STATUS(IR)=-STATIR
  IF(STATIR.EQ.-1)GO TO 40
  LB(IR)=UB(IR)
  UB(IR)=XLARGE
  IF(TOP.EQ.1)GO TO 60
  DO 30 J=1,(TOP-1)
  I=TOP-J
  IF(STACK(I,2).NE.IR)GO TO 30
  IF(STACK(I,1).EQ.-1)GO TO 30
  UB(IR)=STACK(I,3)
  GO TO 60
30 CONTINUE
  GO TO 60
40 UB(IR)=LB(IR)
  LB(IR)=0
  IF(TOP.EQ.1)GO TO 60
  DO 50 J=1,(TOP-1)
  I=TOP-J
  IF(STACK(I,2).NE.IR)GO TO 50
  IF(STACK(I,1).EQ.1)GO TO 50
  LB(IR)=STACK(I,3)
  GO TO 60
50 CONTINUE
60 DO 70 J=1,NA
  IF(LB(J).EQ.UB(J))GO TO 70
  XX=STATUS(J)*A(MC,J)
  IF(XX.GT.EPS)GO TO 80
  CONTINUE
  RETURN
80 ENTER=J
  C1=STATUS(J)
  LEMMIN=XLARGE
  DO 110 I=1,M
  XX=C1*A(I,ENTER)
  IF(ABS(XX).GE.EPS)GO TO 110
  IF(XX.LE.0.)GO TO 90
  LEMDA=-C1*(A(I,NB)-UB(BVAR(I)))/A(I,ENTER)
  GO TO 100
  90 LEMDA=-C1*(A(I,NB)-LB(BVAR(I)))/A(I,ENTER)
100 IF(LEMDA.GE.LEMMIN)GO TO 110
  LEMMIN=LEMDA
110 CONTINUE
  IL=BVAR(LEMPDW)
  XENTER=LB(ENTER)
  IF(C1.EQ.1)XENTER=UB(ENTER)
  X2=UB(ENTER)-LB(ENTER)
  IF(LEMMIN.GE.X2)GO TO 120
  LEMMIN=X2
  STATUS(ENTER)=-C1
120 DO 130 I=1,MC
130 A(I,NB)=A(I,NB)+C1*A(I,ENTER)*LEMMIN
  IF(LEMMIN.EQ.X2)GO TO 170
  STATUS(IL)=-1

```



C

```

IF(A(LEMRDOW, NB).GT.(GB(IL)+EPS))STATUS(IL)=1
A(LEAPDOW, NB)=XENTER-CI*LEMAIN
BVAR(LEMRDOW)=ENTER
STATUS(ENTER)=0
FACTOR=A(LEMRDOW, ENTER)
DO 140 J=1, NA
140 A(LEMRDOW, J)=A(LEMRDOW, J)/FACTOR
DO 150 I=1, MC
IF(I.EQ.LEMRDOW)GO TO 160
TEMP=A(J, ENTER)
DO 150 J=1, NA
150 A(I, J)=A(I, J)-A(LEMRDOW, J)*TEMP
160 CONTINUE
170 IF(ENTER.EQ.1R)GO TO 210
IF(STATUS.EQ.-1)GO TO 190
LB(IR)=0
IF(TOP.EQ.1)GO TO 210
DO 180 J=1, (TOP-1)
I=TOP-J
IF(STATUS(I, 2).NE.1R)GO TO 180
IF(STATUS(I, 1).EQ.1)GO TO 180
LB(IR)=STACK(I, 3)
GO TO 210
180 CONTINUE
GO TO 210
190 UB(IR)=XLARGE
IF(TOP.EQ.1)GO TO 210
DO 200 J=1, (TOP-1)
I=TOP-J
IF(STATUS(I, 2).NE.1R)GO TO 200
IF(STATUS(I, 1).EQ.-1)GO TO 200
UB(IR)=STACK(I, 3)
GO TO 210
200 CONTINUE
210 GO TO 60
END

```

```

C *****
C THIS SUBROUTINE ROUNDS-OFF THE CONTINUOUS VALUED SOLUTION
C IN 2 WAYS AND GIVES THE BEST SOLUTION
C *****
SUBROUTINE ROUNDOFF(RX,IX1,TIX,TC,AMOUNT,USDEM1)
INTEGER SDEM(100),USDEM(100),IX1(100),IX2(100),AMOUNT,TAMUNT,TC
ITCOST,STPOSN(100),DEMAND(100),TUSDEM(100),BVAR(100),STACK(300),
ICL,TOP,STATUS(200),GA(100,100),TIX(100),USDEM1(100)
REAL A(101,200),RX(200),COST(100),GB(200),LB(200)
COMMON A,BVAR,STATUS,GB,LB,NA,NB,MC,EPS,XLARGE,M,N,STACK,TOP,
IGA,STPOSN,SDEM,USDEM,CL,DEMAND
AMOUNT=XLARGE
DO 70 I=1,M
SX=0
TCOST=0
DO 10 J=I+1,I+M-1
J1=J
IF(J.GT.N)J1=J-N
SX1=SX
R2=RX(J1)
K2=R2
IF((R2-K2).LE.EPS)R2=K2
IF((R2-K2).GE.(1-EPS))R2=K2+1
SX=SX1+R2
I1=SX1+(1.-EPS)
I2=SX+(1.-EPS)
IX2(J1)=I2-I1
10 TFCOST=TCOST+IX2(J1)
DO 20 I=1,M
SDEM(I)=0
DO 30 J=1,N
DO 30 J1=STPOSN(J),STPOSN(J)+CL-1
L=I
IF(I.GT.N)L=L-N
IF(GA(L,J).EQ.1)SDEM(I)=SDEM(I)+IX2(J)+TIX(J)
30 CONTINUE
TAMUNT=0
DO 50 I=1,M
TUSDEM(I)=DEMAND(I)-SDEM(I)
IF(TUSDEM(I).GT.0)TAMUNT=TAMUNT+TUSDEM(I)
50 CONTINUE
IF(TAMUNT.GE.AMOUNT)GO TO 70
AMOUNT=TAMUNT
TC=TCOST
DO 60 I=1,M
USDEM1(I)=TUSDEM(I)
IX1(I)=IX2(I)
IF(AMOUNT.EQ.0)GO TO 80
60 CONTINUE
70 RETURN
80 END

```

```

C *****
C ***** THIS SUBROUTINE FINDS THE SOURCE VARIABLE FOR BRANCHING *****
C *****
SUBROUTINE SUBRDX(SV,SR)
  INTEGER BVAR(100),STATUS(200),STACK(300,3),TOP,SR,SV
  INTEGER GA(100,100),IX1(100),TC,STPOSN(100),SDEN(100),USDEM(100)
  IADJUNT,DEMAND(100),CL,TIX(100),USDEM1(100)
  REAL A(101,200),UB(200),LB(200)
  CO44DX A,BVAR,STATUS,UB,LB,WA,NB,PC,EPS,XLARGE,M,N,STACK,TOP,
  IGA,STPOSN,SDEN,USDEM,CL,DEMAND
  IAST=-EPS
  SV=0
  SR=0
  DO 20 J=1,M
    IBV=BVAR(J)
    IF(IBV.GT.0)GO TO 20
    IF(A(J,UB).LT.((IX1(IBV)+EPS)))GO TO 20
    R1=A(J,NB)
    I1=R1+EPS
    I2=R1+(1.-EPS)
    IF(I1.EQ.I2)GO TO 20
    K1=0
    ISF=STPOSN(J)
    DO 10 I11=ISF,ISF+CL-1
      I=I11
      IF(I.GT.M)I=1-M
      IF((GA(I,J).EQ.1).AND.(USDEM(I).GT.0))K1=K1+1
      FRAC=I2-R1
      AM1=K1*FRAC
      IF(AM1.LT.IAMP)GO TO 20
      IAMP=AM1
    SR=J
    SV=IBV
  20 CONTINUE
  RETURN
  END

```

```

C***** DOUBLE PRECISION CUTTING PLANE ALGORITHM *****
C***** DOUBLE PRECISION SUBROUTINE *****
      REAL A(0:200,1:101),DA(100,100),RX(200),ARRFU(200,2)
      REAL ARRFU(20,0:51),MAXRAT
      INTEGER M,N,K,LX(100),POSH(200),COUNT,HIGH,DEMAND(100)
      INTEGER SDEM(100),P,SR,CYCLE(100),CL,AMOUNT
      LOGICAL ITER,FEAS
      COMMON A,M,N
      READ(23,*)NPROB
      DO 1100 I=1,NPROB
        WRITE(24,14)I
        FORMAT(IX,' PROBLEM = ',I2)
        WRITE(24,17)
        FORMAT(SX,' CUTTING PLANE METHOD')
        READ(23,*)M,N,CL
        MD=M
        WRITE(24,30)M,N
        FORMAT(SX,' M = ',I3,' N = ',I3)
        DO 40 J=1,N
          A(0,J)=0
        40 IF(J.LE.N)A(0,J)=1
        READ(23,*)ISEED,LOW,HIGH
        WRITE(24,50)ISEED,LOW,HIGH
        50 FORMAT(SX,' SEED = ',I3,' RANGE OF DEMANDS IS ',I3,' to ',I3)
        READ(23,*)(CYCLE(I),I=1,CL)
        WRITE(24,122)((CYCLE(I),I=1,CL)
        122 FORMAT(SX,' CYCLE: ',36I2)
        CALL GENRAT(ISEED,LOW,HIGH,CYCLE,CL,DEMAND)
        CALL RTIME(N2)
        M=M+N+1
        DO 120 J=1,M
        120 A(M-N,J)=-1
        A(0,N+1)=0
        A(M-N,N+1)=0
        DO 160 I1=M-N+1,M
        DO 160 J1=1,N+1
        160 A(I1,J1)=0.0
        DO 180 I1=M-N+1,M
        A(I1,I1-M+N)=-1
        180 CONTINUE
        DO 200 I=1,M
        IF(I.GE.N)POSN(I)=M-N+1
        IF(I.GT.N)POSN(I)=1-N
        200 CONTINUE
        DO 220 I=1,M-N-1
        DO 220 J=1,N
        220 DA(I,J)=A(I,J)
        EPS=0.0001
        NCUTS=0
        NG=M
        NP=N+1
        R=0
        320 R=R+1
        ITER=A(R,NP) < -EPS
        IF((.NOT.ITER).AND.(R.LT.M))GO TO 320
        IF(.NOT.ITER) GO TO 540
        COUNT=COUNT+1
        ITR=R
        DO 340 N1=ITR+1,M
        IF(A(R,NP).GT.A(N1,NP))R=N1
        340 CONTINUE
        K=0
        380 K=K+1
        ITER=A(R,K) < -EPS
        IF((.NOT.ITER).AND.(K.LT.N)) GO TO 380
        FEAS=ITER
        IF(.NOT.ITER) GO TO 540
        L=K
        IF(K.EQ.N)GO TO 440
        DO 420 J=K+1,N
        IF(A(R,J).GE.-EPS) GO TO 420
        I=-1
        400 I=I+1
        S=(A(I,J)/A(R,J)-A(I,L)/A(R,L)
        IF(S.EQ.0.0) GO TO 400
        IF(S.GT.0.0)L=J
        420 CONTINUE
        TEMP1=A(R,L)
        440 DO 480 I=0,M
        460 A(I,L)=-A(I,L)/TEMP1
        480 DO 520 J=1,NP
        IF(J.EQ.L) GO TO 520
        TEMP=A(R,J)
        DO 500 I=0,M
        500 A(I,J)=A(I,J)+A(I,L)*TEMP
        520 CONTINUE

```

```

540 IF(I/ITER)GO TO 300
    SUMX=-A(0,N+1)
    IF((NDCUTS.EQ.0).AND.(NCUTS.EQ.0))FLPSOL=SUMX
    TYPE *,NDCUTS,NCUTS,SUMX
    ISUMX=SUMX+EPS
    RISUMX=ISUMX
    IF((RISUMX+EPS).GT.SUMX)GO TO 600
    DO 580 J=1,N
    SUM4CUL=0.
    DO 560 I=(NG-N+1),NG
    SUM4CUL=SUM4CUL+A(I,J)
580 A(4G-N,I)=SUM4CUL
    A(4G-N,I+1)=- (RISUMX+1.)-A(0,N+1)
    NCUTS=NCUTS+1
    GO TO 280
600 DO 620 I=4G-N+1,NG
620 RX(I-NG+N)=A(1,N+1)
    A(0,N+1)=-RISUMX
    DO 1000 I11=1,N
    SX=0.
    DO 640 J1=I11,I11+N
    J=J1
    IF(J1.GT.MD)J=J1-MD
    SX1=SX
    SX=SX1+RX(J)
    I1=SX1+0.999
    I2=SX+0.999
640 IX(J)=I2-I1
    AMOUNT=0
    DO 660 J=1,NG-N-1
    SDEM(I)=0
    DO 660 J=1,N
660 IF((A(1,J).EQ.-1.0))SDEM(I)=SDEM(I)+IX(J)
    USDEM=DEMAND(I)-SDEM(I)
    IF(USDEM.GT.0.0)AMOUNT=AMOUNT+1
680 CONTINUE
    IF(AMOUNT.EQ.0)GO TO 900
1000 CONTINUE
    DO 700 I=1,NG
    KR=PDSN(I)
    IP=A(KR,N+1)+EPS
    IF((IP+EPS).LT.(A(KR,N+1))) GO TO 720
    A(KR,N+1)=IP
700 CONTINUE
    GO TO 900
720 M=M+1
    PDSN(M)=M
    P=1
    ARRF0(P,1)=KR
    ARRF0(P,2)=A(KR,N+1)-IP
    DO 740 L1=1,4
    LL=PDSN(L1)
    IP1=A(LL,N+1)+EPS
    IF(IP1.GT.A(LL,N+1))A(LL,N+1)=IP1
    P=P+1
    ARRF0(P,2)=A(LL,N+1)-IP1
    ARRF0(P,1)=LL
740 CONTINUE
    DO 760 K=1,P-1
    DO 760 I=K,P-1
    IF(ARRF0(K,2).GE.ARRF0(I+1,2))GO TO 760
    TEMP1=ARRF0(K,1)
    TEMP2=ARRF0(K,2)
    ARRF0(K,1)=ARRF0(I+1,1)
    ARRF0(K,2)=ARRF0(I+1,2)
    ARRF0(I+1,1)=TEMP1
    ARRF0(I+1,2)=TEMP2
760 CONTINUE
    DO 860 I=1,20
    KR=ARRF0(I,1)
    SUMFIJ=0
    ARRFIJ(I,N+1)=ARRF0(I,2)
    NZFIJ=0
    DO 840 J=1,N
    IP=A(KR,J)
    IF(A(KR,J).LT.-EPS)IP=IP-1
    FIJ=A(KR,J)-IP
    IF(FIJ.GE.EPS)GO TO 780
    FIJ=0
    M1=A(KR,J)
    A(KR,J)=M1
780 IF(FIJ.GT.(1.0-EPS))GO TO 800
    GO TO 820
800 FIJ=0.
    A(KR,J)=IP+1.
820 ARRFIJ(I,J)=FIJ
    SUMFIJ=SUMFIJ+FIJ
    IF(FIJ.NE.0.)NZFIJ=NZFIJ+1
840 CONTINUE

```

```

177000 ARRFLJ(I,0)=RF
178000 ARRFLJ(I,12)=SUMFLJ
179000 IF(SUMFLJ,30.0)GO TO 860
180000 RATIO=ARRFLJ(I,11)/SUMFLJ+NZFLJ
181000 IF(RATIO,LE,MAXRAT)GO TO 860
182000 MAXRAT=RATIO
183000 SR=L
184000 CONTINUE
185000 DO 880 J=1,N+1
186000 A(I,J)=-ARRFLJ(SR,J)
187000 NCUTS=NCUTS+1
188000 IF(NCUTS,GT,60)GO TO 986
189000 GO TO 280
190000 ADINE(21,989)
191000 FORMAT(5X,'THE PROBLEM COULD NOT BE SOLVED AFTER 150 CUTS')
192000 WRITE(24,920)NCUTS
193000 FORMAT(5X,'NO. OF FRACTIONAL CUTS = ',14)
194000 WRITE(24,940)NCUTS
195000 FORMAT(5X,'NO. OF CUTS = ',14)
196000 C WRITE(24,960)(1,IX(I),I=1,N)
197000 FORMAT(5X,'X-',12,14)
198000 ITC=-(A(0,N+1)-EPS)
199000 WRITE(24,980)ITC
200000 FORMAT(5X,'TOTAL COST = ',15)
201000 WRITE(24,984)FLPSOL
202000 984 FORMAT(5X,'FIRST LP SOLUTION = ',F10.3)
203000 CALL RTIME(93)
204000 ICPUT=43-92
205000 WRITE(24,126)ICPUT
206000 126 FORMAT(5X,'TOTAL CPU TIME = ',18,' milli seconds')
207000 WRITE(24,995)
208000 995 FORMAT(//)
209000 1100 CONTINUE
210000 STOP
211000 END
212000 SUBROUTINE GENRAT(ISEED,LOW,HIGH,CYCLE,CL,DEMAND)
213000 READ A(0:200,1:101)
214000 INTEGER CYCLE(100),DIF,HIGH,DEMAND(100),CL
215000 COMMON A,M,N
216000 DO 10 I=1,M
217000 DO 10 J=1,N
218000 10 A(I,J)=0
219000 DIF=M/N
220000 DO 20 J=1,N
221000 K=1
222000 DO 20 I=1,CL
223000 L=1+DIF*(J-1)
224000 IF(L,GT,M)L=L-M
225000 A(L,J)=-CYCLE(K)
226000 K=K+1
227000 20 CALL SETRAN(ISEED)
228000 DO 40 I=1,M
229000 HI=LAN(X)
230000 IDEM=LOW+(HIGH-LOW)*HI+0.5
231000 DEMAND(I)=IDEM
232000 A(I,N+1)=-IDEM
233000 40 CONTINUE
234000 RETURN
235000 END
236000
237000

```



```

12=SA+0.99999
IX(1)=I2-I1
TCOST=TCOST+IX(1)
160 CONTINUE
CALL RAN(02)
K=0
DO 170 I=1,CL
170 IF(CYCLE(I).EQ.1)K=K+1
BOUND=(N+(I-1)/K
IBOUND=BOUND+0.9999
CPUT=I2-I1
WRITE(22,130)N
180 FORCAT(5X,'N = ',14,' M = ',14)
WRITE(22,100)ISEED,LOW,HIGH
190 FORCAT(5X,'SEED = ',15,' RANGE OF DEMANDS IS ',14,' to ',13)
WRITE(22,200)TCOST
200 FORCAT(5X,'TOTAL COST = ',F8.2)
WRITE(22,210)O,K
210 FORCAT(5X,'O = ',12,' K = ',13)
WRITE(22,220)IBOUND
220 FORCAT(5X,'THE BOUND FOR THIS SOLUTION IS ',I6)
WRITE(22,230)CPUT
230 FORCAT(5X,'TOTAL CPU TIME TAKEN = ',15,' milli seconds')
WRITE(22,240)
240 FORCAT(///)
250 CONTINUE
STOP
END
SUBROUTINE GENRAT(ISEED,LOW,HIGH,CYCLE,CL,DEMAND)
REAL A(0:100,0:200)
INTEGER CYCLE(100),DIF,HIGH,DEMAND(100),CL
COMMON A,M,N
DO 10 I=1,M
DO 20 J=1,N
10 A(I,J)=0.
DIF=M/N
DO 20 J=1,N
K=1
DO 20 I=1,CL
L=1+DIF*(I-1)
IF(L.GT.M)L=M-N
A(L,J)=-CYCLE(K)
20 K=K+1
CALL SETRAN(ISEED)
DO 30 I=1,M
U1=RAN(X)
IDEM=(LOW+(HIGH-LOW)*U1+0.5
DEMAND(I)=IDEM
A(I,J)=-IDEM
30 CONTINUE
RETURN
END

```



**A 87610**